HP-UX 参考手册

第 5 节：其他主题

HP-UX 11i v3

第 9 卷 （共 10 卷）



生产部件号：B2355-91045

E0207

法律声明

本文档中的信息如有更改，恕不另行通知。

担保

随 HP 产品及服务提供的明示性担保声明中列出了适用于此 HP 产品及服务的专用担保条款。本文中的任何内容均不构成额外的担保。 HP 对本文中的技术或编辑错误以及缺漏不负任何责任。

美国政府许可

机密计算机软件。必须有 HP 授予的有效许可证，方可拥有、使用或复制本软件。根据供应商的标准商业许可证的规定，美国政府应遵守 FAR 12.211 和 12.212 中有关 “商业计算机软件”、“计算机软件文档” 与 “商业货物技术数据” 条款的规定。

附加版权声明

本文档及其所涉及的软件可能同时受到下述一项或多项版权的保护。某些单独的联机帮助页将对这些附加版权加以认可。

商标声明

Intel 和 Itanium 均为 Intel Corporation 在美国和其他国家（地区）的注册商标，使用时已受到许可。

Java 是 Sun Microsystems, Inc. 在美国的商标。

MS-DOS 和 Microsoft 是 Microsoft Corporation 在美国的注册商标。

OSF/Motif 是 The Open Group 在美国和其他国家 （地区）的商标。

UNIX 是 Open Group 的注册商标。

X Window System 是 X/Open Group 的商标。

前言

HP-UX 是 Hewlett-Packard Company 开发的一种 UNIX® 操作系统，可与各种行业标准兼容。该操作系统基于 System V Release 4 操作系统，并包括 Fourth Berkeley Software Distribution 中的重要功能。

本手册共十卷，由系统参考文档资料组成，其中包括若干个称为**联机帮助页**的条目，这些联机帮助页以 man 命令来命名，通过该命令，可以在系统上显示这些联机帮助页条目（请参阅 *man* (1)）。这些条目也称为联机帮助页和参考页。

简介

有关 HP-UX 以及联机帮助页的结构和格式的简介信息，请参阅第 9 卷中的 *introduction* (9) 联机帮助页。

小节简介

联机帮助页分为若干节，各节也包含 introduction 或 intro 联机帮助页，这两个联机帮助页对相应的节中的具体内容进行了说明。这些节有：

intro (1)	**第 1 节：用户命令** （第 1 卷包含 A-M ；第 2 卷包含 N-Z）
intro (1M)	**第 1M 节：系统管理命令** （第 3 卷包含 A-M ；第 4 卷包含 N-Z）
intro (2)	**第 2 节：系统调用** （第 5 卷）
intro (3C)	**第 3 节：库函数** （第 6 卷包含 A-M ；第 7 卷包含 N-Z）
intro (4)	**第 4 节：文件格式** （第 8 卷）
intro (5)	**第 5 节：其他主题** （第 9 卷）
intro (7)	**第 7 节：设备专用文件** （第 10 卷）
intro (9)	**第 9 节：常规信息** （第 10 卷）
索引	**所有卷中的索引** （第 10 卷）

印刷字体约定

audit (5)	表示 HP-UX 联机帮助页。“audit”是联机帮助页名称，“5”是该帮助页在《HP-UX 参考手册》中的小节号。在网站和 Instant Information DVD 上，可能是指向该联机帮助页的热链接。在 HP-UX 命令行输入“`man audit`”或“`man 5 audit`”可以查看该联机帮助页。详见 *man* (1)。
《书名》	表示文档中引用的书籍、手册的名称，以宋体表示。
“术语”	表示文档中引用的专用术语，以宋体表示。
键盘操作	键盘键名称。注意，Return 和 Enter 指的是同一个键。
强调内容	第一次定义的名词和强调的内容用**黑体**表示。
`系统字体`	表示计算机显示的文本和系统项。
可替换变量	命令、功能中可以替换的变量名以仿宋表示。
[ ]	格式和命令说明中的可选内容。如果内容用“|”分隔，就必须选择其中之一。
{ }	格式和命令说明中的必需内容。如果内容用“|”分隔，就必须选择其中之一。
....	前面的元素可以重复任意多次。
|	分隔选项列表中的项目。

命令语法

文本	可按原样输入的词或字符。
可替换变量	用适当的值来替换的词或字符。
intro1	分为一组的一个或多个命令选项 -ikx。 chars 通常是由一系列字符组成的字符串，每个字符表示一个具体选项。例如， -ikx 等效于单独列出的选项 -i、 -k 和 -x。有时使用加号 (+) 作为选项前缀。
intro1M	一个命令选项，例如， -help。 word 是一个关键字。与 -chars 的区别通常是显而易见的，在 “选项”说明中对其进行了 解释。有时使用加号 (+) 和双连字符 (--) 作为选项前缀。
[ ]	格式和命令描述中可选的内容。
{ }	格式和命令描述中必需的内容。
|	竖线用来分隔一系列选项中的各个项，通常用在方括号或花括号中。
...	标记后面的省略号 (abc...)、右方括号后面的省略号 ([ ]...) 或右花括号后面的省略号 ({ }...) 表示前面的元素及其前面的空白字符 （如果有）可重复若干次。
...	省略号有时表示一个范围内省略的项。

函数概要和语法

HP-UX 函数采用定义格式，而不是用法格式。定义格式包含类型信息，在程序中插入此函数调用时会省略该信息。

函数的语法元素与命令的语法元素相同，但选项除外；请参阅第 VII 页上的“命令语法”。

函数常规定义

常规定义格式为：

类型　函数（类型　参数［，类型　参数］．．．）；

例如：

```
int setuname ( const char *name , size_t namelen );
```

函数用法

用法格式为：

函数（参数［，参数］．．．）；

例如：

```
setuname ( name [ , namelen ]... );
```

版本说明

部件号	发行版；日期；文档格式；发布形式
B2355-91037~46	HP-UX 11i v3；2007 年 2 月；PDF （10 卷）；docs.hp.com 和印刷版本。中文版。
B2355-90931~40	HP-UX 11i v2；2004 年 9 月；PDF （10 卷）；docs.hp.com 。中文版。

第 9 卷

目录

第 5 节

第 9 卷

目录

第 5 节

目录
第 9 卷

第 5 节：其他主题

条目名称(节)：名称 说明

目录
第 9 卷

条目名称(节)：名称 说明

目录
第 9 卷

条目名称(节)：名称 说明

目录
第 9 卷

名称

intro - 杂项简介

说明

本节介绍了杂项工具，例如：宏程序包、字符集表、文件系统层次结构和操作系统可调参数。

另请参阅

introduction(9)。

其他文档

- 可调内核参数
- 在 Internet 上 **http://docs.hp.com** 站点的 HP-UX。

名称

acctsuspend、acctresume - 当可用磁盘空间达到阈值时挂起并恢复记账

值

无故障

缺省值。

缺省值

acctsuspend： **2**

acctresume： **4**

允许值

acctresume： **-100 - 101**

acctsuspend： **-100 - acctresume-1**

建议值

acctsuspend： **2 - 6**

acctresume： **4 - 10** （但是大于 acctsuspend）

说明

acctsuspend 和 **acctresume** 可调参数控制由于磁盘空间限制而在何时停止和恢复记账。当记账使用的文件系统可用磁盘空间达到挂起阈值，此阈值为相对于仅超级用户可用的磁盘空间百分比的 **acctsuspend** 百分比时，记账将被挂起，直到可用磁盘空间达到恢复阈值为止，此阈值为相对于仅超级用户可用的磁盘空间百分比的 **acctresume** 百分比。

> 注释：由于 **acctsuspend** 和 **acctresume** 值是相对于仅超级用户可用的磁盘空间百分比而指定的，因此这些参数的负值可以有意义。例如，如果超级用户在文件系统创建时保留了文件系统百分之十的磁盘空间，并且 acctsuspend 为 -5，同时 acctresume 为 0，则挂起阈值将为总磁盘空间的百分之五，而恢复阈值将为总磁盘空间的百分之十。

应该由谁来更改此可调参数？

使用记账的任何人。

对于更改的限制

对此可调参数的更改将在下次重新引导时生效。

何时应增加此可调参数的值？

当有必要在计账文件系统上维持较高的可用空间百分比时，应该考虑增加任一变量。

增加此可调参数的值的副作用是什么？

任一值越高，捕捉到的记账数据越少。这些值分隔得越远，可能丢失的记账数据越多。

何时应降低此可调参数的值?

如果记账数据需要额外的磁盘空间，且它无法通过从文件系统删除文件来获得，那么应该降低 **acctsuspend** 的值。

降低此可调参数值的副作用是什么?

当文件系统已满时，文件系统性能（写入记账记录）降低。依次，这样会降低记账进程的整体性能。

同时还应更改哪些其他可调参数的值?

更改这些可调参数之一时，两个可调参数都需要考虑。

警告

所有 HP-UX 内核可调参数都是针对于特定版本的。未来的 HP-UX 版本可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

acctsuspend 和 **acctresume** 由 HP 开发。

另请参阅

accton(1M)。

名称

acl - HFS 访问控制列表简介

说明

访问控制列表是自由访问控制的关键执行机制（请参阅下文的“定义”一节），它比传统的 HP-UX 机制指定的用户和组对文件的访问具有更强的可选择性。

HP-UX 已经允许无权限用户或进程（如文件的所有者），在“需要知道”的基础上允许或拒绝其他用户对文件和其他对象的访问，就像通过它们的用户和（或）组的标识决定一样（请参阅 *passwd*(4) 和 *group*(4) ）。要实现此级别的控制，可以通过设置或操作文件的权限位来授权或限制所有者、组和其他用户的访问（请参阅 *chmod*(2)）。

ACL 比权限位具有更强的可选择性。 ACL 允许文件所有者或超级用户允许或拒绝访问用户、组或其组合的列表。

ACL 仅可针对文件用作 UNIX 操作系统上的自由访问控制 (DAC) 机制的超集，而对于诸如进程间通信 (IPC) 对象等其他对象来说，则不支持此功能。

此联机帮助页描述的 ACL 仅能运行在 HFS 文件系统上。有关 JFS 文件系统中各 ACL 的描述，请参阅 *aclv*(5) 。

定义

因为数据访问控制是计算机安全的关键部分，我们根据《Department of Defense Trusted Computer System Evaluation Criteria》提供下列定义，以便更好地解释访问控制的概念以及它与 HP-UX 安全特性的联系：

访问
: “一个主体与一个对象特定类型的交互使得信息由一端流向另一端。”主体包括“人、进程或能让信息在对象间流动或改变系统状态的设备”。对象包括文件（普通文件、目录、特殊文件以及 FIFO 等）和进程间通信 (IPC) 功能（共享内存、消息队列、信号量、套接字）。

访问控制列表 **(ACL)**
: 访问控制列表是与为所有可能的用户 ID/ 组 ID 组合指定访问权限的文件相关联的 (*user.group, mode*) 条目集合。

访问控制列表 **(ACL)** 条目
: ACL 中的条目，用于为一个用户和组的 ID 组合指定访问权限。

更改权限
: 更改 DAC 信息（权限位或 ACL 条目）的权限。将更改权限授予对象（文件）所有者和具有权限的用户。

自由访问控制 **(DAC)**
: “基于主体和（或）其从属的组的标识，对对象的访问权限进行限制的一种方法。这种控制的自由性体现在一个具有某种访问权限的主体能够将其权限（可能是间接的）传递给其他主体。”

模式
: 每个 ACL 条目中有三个位，分别代表读取、写入和执行/搜索权限。除了与文件系统的每一个文件相关联的 16 模式位以外，这些位也可能会存在（请参阅 *glossary*(9) ）。

权限　　忽略访问限制以及更改限制的功能，这些限制是由安全策略所强加的，并在一种访问控制机制中实现。在 HP-UX 中，仅超级用户和特定组的成员（请参阅 *privgrp*(4) ）是特权用户。

限制条目与许可条目
　　根据相关环境可以判断单个 ACL 条目是许可条目还是限制条目。限制条目将拒绝用户和（或）组的访问，否则，根据常规控制或可选 ACL 条目（请参阅下文），他们可能会具有该访问权限。许可条目为用户和（或）组授权，否则，根据常规控制或可选 ACL 条目，他们可能会被拒绝访问。

访问控制列表条目

访问控制列表 (ACL) 由多组与指定权限的文件相关联的 (*user.group, mode*) 条目组成。每一个条目为一个用户 ID/组 ID 组合指定一组访问权限，其中包括读取、写入和执行/搜索。

为了帮助理解访问控制列表和传统的文件权限之间的关系，请考虑下面的文件及其权限：

-rwxr-xr--　james　admin　datafile

文件所有者是用户 **james** 。
文件的组是 **admin** 。
文件名是 **datafile** 。
文件所有者权限是 **rwx** 。
文件组权限是 **r-x** 。
文件的其他权限是 **r--** 。

在 ACL 中，用户 ID 和组 ID 可由 **/etc/passwd** 中的名称或编号表示。也可使用下列特殊符号：

%　代表非特定用户或组的符号。
@　表示当前文件所有者或组的符号。

基本 ACL 条目

当一个文件创建以后，将从文件的访问权限位映射三个基本访问控制列表条目，以便匹配一个文件的所有者和组，以及文件的传统权限位。基本 ACL 条目可由 *chmod*(2) 和 *setacl*(2) 系统调用来更改。

(*uid*.%,mode)　文件所有者的基本 ACL 条目
(%.*gid*,mode)　文件组的基本 ACL 条目
(%.%,mode)　其他用户的基本条目

（除了注释的地方之外，各示例都以短格式表示法表示。请参阅下文的“ACL 表示法”一节）。

可选的 ACL 条目

可选的访问控制列表条目包含附加的访问控制信息，这些信息可由用户使用 *setacl*(2) 系统调用来设置，从而进一步允许或拒绝文件访问。最多可指定 13 个附加的用户/组的组合。

例如，下列可选的访问控制列表条目可与文件相关联：

(mary.admin, rwx) 为组 **admin** 中的用户 **mary** 授予读取、写入和执行权限。

(george.%, ---) 拒绝非特定组中的用户 **george** 的任何访问。

ACL 表示法

受支持的库调用和管理 ACL 的命令识别三种不同的符号表示形式：

运算符格式 对于整个 ACL 的输入和对现有 ACL 的修改，使用一种与 *chmod*(1) 使用的语法相似的语法。

短格式 主要是对于输出来说更易于读取。*chacl*(1) 接受此格式作为输入，使其可解释 *lsacl*(1) 的输出。

长格式 多行格式对提高清晰度非常有用，且只有输出支持此格式。

对于示例文件，基本的 ACL 条目可以下列三种表示法表示：

运算符格式	james.% = rwx, %.admin = rx, %.% = r	
短格式	(james.%,rwx) (%.admin,r-x) (%.%,r--)	
长格式	rwx	james.%
	r-x	%.admin
	r--	%.%

除了基本的 ACL 用法外，某些库调用和命令理解并使用一种运算符变化形式和短格式。请参阅下文的“ACL 模式”一节。

ACL 唯一性

每一个 ACL 中的条目都是唯一的。对于任意一对 *u* 和 *g* 值，只能有一个 (*u.g, mode*) 条目；对于一个给定的 *u* 值，只能有一个 (*u.%, mode*) 条目；对于一个给定的 *g* 值，只能有一个 (*%.g, mode*) 条目，而对于每一个文件，则只能有一个 (*%.%, mode*) 条目。例如，一个 ACL 可有一个 (23.14, *mode*) 条目和一个 (23.%, *mode*) 条目，但不能有两个 (23.14, *mode*) 条目或两个 (23.%, *mode*) 条目。

检查访问权限算法

ACL 条目可按四个级别的特征分类。在访问权检查中，各 ACL 按此顺序与有效用户 ID 和组 ID 进行比较：

(*u.g*, rwx) 特定用户，特定组
(*u*.%, rwx) 特定用户，非特定组
(%.*g*, rwx) 非特定用户，特定组
(%.%, rwx) 非特定用户，非特定组

一旦进程有效用户 ID 和有效组 ID （或任何补充组 ID）组合的一个条目与之相匹配，则不再进一步检查其他（即，常规的）条目。更多匹配的特定条目要优先于也匹配的不太特定的条目。

如果一个进程有多个组 ID （即，一个非空的补充组列表），则可能会有多个 (*u.g, mode*) 或 (*%.g, mode*) 条目申请此进程。如果是这样，则所有匹配的条目（具有相同的特征级别 *u.g* 或 *%.g*) 中的访问模式将一起进行 OR 运算。如果所得的模式位允许访问，则授予此访问权。因为条目都是唯一的，所以每一个条目类型中的条目顺序无关紧要。

由于传统的 UNIX 权限位映射到基本 ACL 条目中，因此它们包括在访问权检查中。

如果发出请求需要多种访问类型，如打开一个文件供读取和写入，则仅当此进程允许有所有请求的访问类型时，才会授予此访问权。请注意，如果进程的组列表中有两个组，其中一个组仅允许进行读取访问，而另一个仅允许进行写入访问，则可授予此访问权。换句话说，即使是任意一个条目都没有授予请求的访问权，此访问权也可由一个条目组合来授予，因为此进程属于多个组。

ACL 的运算符格式（仅适用于输入）

*user***.** *group operator mode* [*operator mode*]... , ...

多个条目由逗号分隔，与 *chmod*(1) 中的一样。每一个条目包含一个用户标识符和组标识符，其后跟有一个或多个运算符和模式字符，这与 *chmod*(1) 所接受的模式语法相同。

整个 ACL 必须是一个参数，因而，如果它包含空白字符或特殊字符，则应使用引号将其括到 Shell 中。空白字符将被忽略，但名称中的除外。一个空的 ACL 是合法的，它表示“无法访问”或“无法更改”，视相关环境而定。

每一个用户 ID 或组 ID 可由下列形式表示：

name 有效的用户名或组名。
number 有效的数字 ID 值。
% “非特定用户或组，”（如果可行）。
@ “当前文件所有者或组，”（如果可行）；有助于引用一个文件的 *u.%* 和 *%.g* 基本 ACL 条目。

每一个条目中总是需要一个运算符。这些运算符是：

= 将条目中的所有位设置为给定的模式值。
+ 设置条目中指定的模式位。
- 清除条目中指定的模式位。

此模式由 **0** 到 **7** 的八进制值表示；或可以任意顺序指定 **r**， **w** 和 **x** 的任意组合（请参阅下文的“举例”一节）。如果运算符是 **=** ，则一个空模式将拒绝访问；或如果运算符是 **+** 或 **-** ，则空模式表示“无法更改”。

以指定的顺序应用多个条目和一个条目中的多个运算符模式部分。冲突不会导致出错；最后指定的条目或运算符有效。条目无须按任何特定的顺序显示。

请注意， *chmod*(1) 只允许 **u** 、 **g** 、 **o** 或 **a** 以符号的形式分别引用文件所有者、组、其他或所有用户。由于 ACL 与任意用户和组标识符一同使用，因此 @ 是为了方便起见而提供的。

确切的语法是：

acl ::= [*entry*[,*entry*]...]
entry ::= *id* . *id op mode* [*op mode*]...
id ::= *name* | *number* | % | @
op ::= = | + | -
mode ::= 0..7 | [*char*[*char*]...]
char ::= r | w | x

ACL 的短格式（适用于输入和输出）

(*user* **.** *group***,** *mode***)** *...*

短格式与运算符格式在几个方面都有不同：

- 各条目由括号括起来，而不是由逗号分隔。
- 每一个条目都指定模式，其中包括所有的模式位。不可能使用 **+** 和 **-** 运算符来更改此模式值。然而，逗号的功能类似于运算符格式中的 **=** 运算符。
- 为了清晰起见，连字符代表模式字段的输出中未设置的权限位，并且允许在输入中使用。这类似于由 *ls*(1) 使用的模式输出样式。

多个条目连接在一起。为了与运算符格式保持一致，使用一个点 **(.)** 来分隔用户和组 ID。

输出时，不打印空白字符，但名称中的除外（如果有的话）。如果没有匹配的名称是可知的，则打印 ID 号。任一个 ID 都可打印为“非特定用户或组”的 **%** 。模式表示为 **<r|-><w|-><x|->**，即，它始终有三个字符，并使用连字符填充未设置的模式位。如果 ACL 是从系统读取的，则各条目按特征排序，然后按 ID 部分的数字值排序。

输出时，整个 ACL 必须是一个参数，因此，如果它包含空白字符或特殊字符，则应使用引号将其括到 Shell 中。空白字符将被忽略，但名称中的除外。一个空的 ACL 是合法的，它表示“无法访问”或“无法更改”，视相关环境而定。

用户和组 ID与运算符格式的表示一致。

此模式由 **0** 到 **7** 的八进制值表示；或可以任意顺序指定 **r** 、 **w** 、 **x** 和 **-** （忽略）的任意组合（请参阅下文的“举例”一节）。空模式将拒绝访问。

冗余不会导致出错；任意一个用户 ID/组 ID 组合的最后一个条目生效。条目无须按任何特定的顺序显示。

确切的语法是：

acl ::= [*entry*[*entry*]...]
entry ::= (*id*.*id*,*mode*)
id ::= *name* | *number* | % | @
mode ::= 0..7 | [*char*[*char*]...]
char ::= r | w | x | -

ACL 的长格式（仅适用于输出）

mode user **.** *group*

每个条目在输出中单独占一行。模式首先显示在一个固定宽度的字段中，为了方便地进行纵向扫描，此模式使用连字符（对于未设置的模式位）。如果每一个用户和组 ID 都是可知的，则将其显示为一个名称；如果是未知的，则显示为一个数字或“非特定用户或组”的 **%** 。各条目按特定程度的顺序从高到低排序，然后按 ID 部分的数字值排序。

请注意，每一个输出的 ACL 至少有三个条目，即基本 ACL 条目（即，*uid*.%、%.*gid* 和 %.%）。

确切的语法是：

acl ::= *entry*[<*newline*>*entry*]...
entry ::= *mode*<*space*>*id.id*
mode ::= <rl-><wl-><xl->
id ::= *name* | *number* | %

ACL 模式

某些库调用和命令识别和使用 ACL 模式，而不是确切的 ACL，从而允许在匹配此模式的所有条目上进行操作。ACL 语法按下述方式扩展：

通配符用户 ID 和组 ID

***** （通配符）的一个用户名或组名匹配任意一个条目中的用户 ID 或组 ID，其中包括 **%** （非特定用户或组）。

打开、关闭或忽略模式位

对于运算符格式的输入，运算符 **=** 、 **+** 和 **-** 应用如下：

= 条目模式值完全匹配此模式值
+ 在条目模式值中打开的这些位
- 在条目模式值中关闭的这些位

当仅使用 **+** 和 **-** 运算符时，命令忽略未指定的模式位的值。

短格式模式对模式和对运算符格式中的 **=** 运算符同等对待。

通配符模式值 只要未在运算符格式的条目中指定其他模式值，则运算符或短格式输入中的 ***** （通配符）模式（例如，“ajs.%=*”或“(ajs.%,*)”）匹配任意一个模式值。而且，可同时省略条目中的模式部分来得到相同的效果。

未组合条目 未组合用户 ID 值和组 ID 值都匹配的条目。接受模式的命令分别应用每一个指定的条目。

支持的 ACL 操作

系统调用 *setacl*(2) 和 *getacl*(2) 允许格式为 *acl_entry* 结构数组的文件设置或获取整个 ACL。要检查对一个文件的访问权限，请参阅 *access*(2) 和 *getaccess*(2) 。

提供用来管理 ACL 的各种库调用：

acltostr(3C) 将 *acl_entry* 数组转换为可打印的字符串。

strtoacl(3C) 解析 ACL 字符串，并将其转换为 *acl_entry* 数组。

strtoaclpat(3C) 解析 ACL 模式字符串，并将其转换为 *acl_entry_patt* 数组。

setaclentry(3C)
fsetaclentry 在一个文件的 ACL 中添加、修改或删除单个 ACL 条目。

cpacl(3C)
fcpacl 将一个 ACL 和文件的其他模式位（请参阅 *chmod*(2) ）从一个文件复制到另一个文件，如有需要，还将转移所有权（请参阅下文），并正确处理远程文件。

chownacl(3C) 更改以一个 ACL 来表示的文件所有者和（或）组，即，转移所有权（请参阅下文）。

可使用下列命令来管理 ACL 和权限：

chacl(1) 添加、修改或删除一个或多个文件上各 ACL 中的单独条目或所有可选条目，删除文件的所有访问权，或将多个 ACL 合并到权限位中。

lsacl(1) 列出文件上的 ACL。

chmod(1) 更改权限位和其他文件的其他模式位。

ls(1) 以长格式列出权限位和其他文件属性。

find(1) 根据文件的属性查找文件（包括 ACL）。

getaccess(1) 列出文件的访问权限。

ACL 与 **stat**、**chmod** 和 **chown** 的交互

stat *st_mode* 字段汇总了调用方对文件的访问权限。仅当文件具有一个或多个对调用方可行的可选条目时，它才与文件权限位不同。 *st_basemode* 字段提供文件的实际权限位。 *st_acl* 字段指示在文件的 ACL 中存在可选 ACL 条目。

st_mode 字段包含一个用户相关的汇总信息，所以，对使用 *stat*(2) 和 *chmod*(2) 的 ACL 不了解的程序更有可能产生预期的结果，而且 *stat*(2) 提供关于 NFS 上远程文件的合理信息。 *st_basemode* 和 *st_acl* 字段仅对于本地文件有用。

chmod 为了符合 IEEE Standard POSIX 1003.1-1988， *chmod*(2) 删除一个文件的 ACL 中所有的可选条目。遗憾的是，由于 *chmod*(2) 用于设置文件其他的模式位和权限位，在某些情况下保留文件的 ACL 需要进行特殊处理。

chown 如果一个文件的新的所有者和（或）组在此文件的 ACL 中还没有一个可选的 (*u.%, mode*) 和（或）(*%.g, mode*) 条目，那么它将继承旧的所有者和（或）组的文件访问权限位和基本 ACL 条目：

(*id1,mode1*) -> (*id2,mode1*)

这是传统行为。然而，如果一个文件的新的所有者和（或）组在此文件的 ACL 中已经有一个可选的 (*u.%, mode*) 和（或）(*%.g, mode*) 条目，那么此 ACL 将不改变：

(*id1, mode1*) -> (*id1, mode1*)
(*id2, mode2*) -> (*id2, mode2*)

ACL 中的现有访问信息保留。然而，因为旧的可选 ACL 条目变成新的基本 ACL 条目（反之亦然），所以文件的访问权限位改变。

利用 *chown*(2) 转移 ACL 的所有权，允许将一个文件传递给不同的用户或组，或由不是所有者的不同用户或组对其进行复制（使用 *cpacl*(3C) 或 *chownacl*(3C)），并且稍后将其返回给原始所有者或组而不对其 ACL 进行网络更改。多余的复杂度是非常必要的，因为：

- ACL 是权限位（连接到文件的所有者和组 ID）的一个向后兼容超集，它们不对这些权限位进行替换。
- 它允许那些处理 ACL 的用户和程序仅需这样操作，而不必处理权限位与 ACL条目的组合。而且，检查访问权限算法更简单且更对称；权限位不“遮蔽”或“隐藏” ACL 条目。

举例

运算符格式

下述示例设置 **%.%** 条目来限制“其他”用户仅读取文件。

chacl '%.% = r' myfile

下述示例允许任一组中的用户“bill”写入文件，同时假定没有限制的条目比 **bill.%** 条目更具有特定性（例如，**bill.adm** 条目拒绝写入）。

chacl 'bill.% +w' myfile

下述 ACL 规范包含两个条目。第一个条目为组 4 中的用户 12 删除对此条目的写入功能，并添加读取功能。第二个条目拒绝任一未指定组中任一未指定用户的访问。

chacl '12.4-w+r, %.% =' myfile

下述一对条目为文件的所有者设置 *u.%* 条目，从而允许其读取和执行文件，而且这样将导致为“其他”用户（“*%.%*”条目）添加读取和执行功能。请注意，此处有意重复了一个模式字符，以便说明。

chacl '@.% = 5, %.% + xwx' myfile

短格式

这里是一个典型的可能会输出的 ACL。它允许当用户 **jpc** 在组 **adm** 中的时候读取或执行此文件；它拒绝当用户 **ajs** 在组 **trux** 中的时候访问此文件；它允许当用户 **jpc** 在任一组（ **adm** 除外）的时候仅读取此文件；组 **bin** 中的其他任意用户可读取或执行此文件；而其他任意用户可能仅可读取此文件。

(jpc.adm,r-x)(ajs.trux,---)(jpc.%,r--)(%.bin,r-x)(%.%,r--)

下述示例允许“其他”用户仅读取此文件。

chacl '(%.%,r)' myfile

下述示例为任一组中的用户 **bill** 设置仅写入的访问权。

chacl '(bill.%,-w-)' myfile

下述示例为组 4 中的用户 12 设置条目，从而允许其读取和写入文件。

chacl '(12.4,wr)' myfile

下述示例为文件的所有者设置基本 ACL 条目，从而允许其读取和执行文件，并且为“其他”用户（“*%.%*”条目）设置读取和执行功能。

chacl '(@.%, 5) (%.%, xwx)' myfile

长格式

这里是与前面示例相同的且以长格式输出的 ACL。

```
r-x    jpc.adm
---    ajs.trux
r--    jpc.%
r-x    %.bin
r--    %.%
```

ACL 模式

下述命令查找多个文件的位置，这些文件中的 ACL 包含一个条目，此条目允许某些用户/组的组合进行读取访问，并拒绝它们进行写入访问。

find / -acl '*.*+r-w' -print

下述示例为组 **bin** 中的任意一个用户，以及任一组中的用户 **tammy** 匹配条目，无论这些条目的模式值是什么。匹配的可选 ACL 条目被删除，匹配的基本 ACL 条目中的模式值设置为零。

chacl -d '%.bin, tammy.*=*' myfile

下述示例匹配所有条目，同时删除可选条目，并将基本 ACL 条目的模式值设置为零。

chacl -d '(*.*,*)' myfile

头文件

头文件 **<sys/acl.h>**

<sys/acl.h> 头文件定义下列常量，以便控制每一个 ACL 的条目数：

NACLENTRIES	每一个 ACL 的最大条目数（包括基本条目）
NBASEENTRIES	基本条目数
NOPTENTRIES	可选条目数

还定义 ACL 条目结构 **struct**acl_entry ，并包括下述成员：

```
aclid_t    uid;     /* user ID */
aclid_t    gid;     /* group ID */
aclmode_t  mode;    /* see <unistd.h> */
```

<sys/acl.h> 头文件还定义类型 **aclid_t** 和 **aclmode_t** 。

非特定用户 ID 值和组 ID 值：

ACL_NSUSER	非特定用户 ID
ACL_NSGROUP	非特定组 ID

一个特殊的 *nentries* 值 **ACL_DELOPT** 与 *setacl*(2) 结合使用来删除可选条目。

头文件 **<sys/getaccess.h>**

<sys/getaccess.h> 头文件定义与 *getaccess*(2) 结合使用的常量。

uid 的特殊参数值:

UID_EUID	使用有效用户 ID
UID_RUID	使用实际用户 ID
UID_SUID	使用保存的用户 ID

ngroups 的特殊参数值:

NGROUPS_EGID	进程的有效 gid
NGROUPS_RGID	进程的真实 gid
NGROUPS_SGID	进程保存的 gid
NGROUPS_SUPP	仅是进程的补充组
NGROUPS_EGID_SUPP	进程的有效 gid 加补充组
NGROUPS_RGID_SUPP	进程的真实 gid 加补充组
NGROUPS_SGID_SUPP	进程保存的 gid 加补充组

头文件 **<acllib.h>**

<acllib.h> 头文件定义几个与ACL 支持库调用结合使用的常量

acltostr() 的 ACL 符号格式:

FORM_SHORT
FORM_LONG

各种调用的 Magic 值:

ACL_FILEOWNER	文件的所有者 ID
ACL_FILEGROUP	文件的组 ID
ACL_ANYUSER	通配符用户 ID
ACL_ANYGROUP	通配符组 ID
MODE_DEL	删除一个 ACL 条目

ACL 条目中有效模式位的掩码:

MODEMASK (R_OK | W_OK | X_OK)

<acllib.h> 头文件还定义 **struct acl_entry_patt** ACL 模式条目结构, 此结构包括下述成员:

```
aclid_t     uid;      /* user ID */
aclid_t     gid;      /* group ID */
aclmode_t   onmode;   /* mode bits that must be on */
aclmode_t   offmode;  /* mode bits that must be off */
```

警告

ACL 用于普通文件和目录。在由某些系统实用程序如终端特殊文件和 LP 调度程序控制文件所处理的文件中，不推荐使用可选 ACL 条目。这些实用程序可能会删除可选条目（包括那些目的具有限制性的条目），而不发送警告作为调用 *chmod*(2) 的结果，从而无法预知地增加了访问。

支持的大多数而并非全部实用程序可以正确处理多个 ACL。然而，只有 *fbackup*(1M) 和 *frecover*(1M) 文件归档实用程序可正确处理访问控制列表。当使用程序（如归档程序 *ar*(1) 、 *cpio*(1) 、 *ftio*(1) 、 *tar*(1) 和 *dump*(1M) ）而无法处理具有可选 ACL 条目的文件的各 ACL 时，请注意在它们各自参考页上包括的访问控制列表信息，以避免丢失数据。

如果在 **/etc/passwd** 文件中已定义一个用户名，或在 **/etc/group** 文件中已将一个组名定义为 **%** 、@ 或模式 *，则 ACL 语法无法按此名称自身来引用，因为此符号具有其他意义。然后，通过 ID 编号仍可对这样的用户或组进行引用。用户名和（或）组名禁止包括下列符号：

- **.** 不在用户名中使用。
- **+** 不在组名中使用。
- **-** 不在组名中使用。
- **=** 不用于组名的运算符格式输入。
- **,** 不用于短格式或运算符格式模式。
- **)** 不用于短格式模式。

有可能使用 @ （文件所有者或组）或 * （通配符）符号来指定一个 ACL 模式，所以有可能根据某些文件的所有权，此模式无法与其相匹配。模式的指定方法是给出两个条目，一个具有特定的值，另一个使用 @ 或 * ，两个条目对于一个文件来说是等效的，但它们包含不同的模式值。例如：

find / -acl '(ajs.%,r)(@.%,rw)' -print

无法匹配由 **ajs** 所拥有的文件。

相关内容

NFS　NFS 不支持远程文件上的 ACL。单独的手册条目指定在这些情况下的各种系统调用、库调用和命令的行为。当在网络上传输具有可选条目的文件，或处理一个远程文件的时候，要谨慎小心，因为这些可选条目可能会以无提示方式删除。

作者

这里描述的访问控制列表设计由 HP 开发。

文件

<sys/acl.h>	支持 *setacl*(2) 和 *getacl*(2) 的头文件。
<sys/getaccess.h>	支持 *getaccess*(2) 的头文件。
<acllib.h>	支持 ACL 库调用的头文件。
/etc/passwd	定义用户名、用户 ID 值和组 ID 值。
/etc/group	定义组名。

另请参阅

chacl(1)、 chmod(1)、 cp(1)、 find(1)、 getaccess(1)、 ln(1)、 ls(1)、 lsacl(1)、 mv(1)、 rm(1)、 fbackup(1M)、 frecover(1M)、 fsck(1M)、 fsdb(1M) access(2)、 chmod(2)、 chown(2)、 creat(2)、 getaccess(2)、 getacl(2)、 mknod(2)、 open(2)、 setacl(2)、 stat(2)、 acltostr(3C)、 chownacl(3C)、 cpacl(3C)、 setaclentry(3C)、 strtoacl(3C)、 group(4)、 passwd(4)、 privgrp(4)、 aclv(5)。

名称

aclv - JFS 访问控制列表 (ACL) 简介

说明

访问控制列表 (ACL) 是自由访问控制（请参阅下文的“定义”一节）的关键执行机制，它比传统的 HP-UX 机制所指定的用户和组对文件的访问具有更强的可选择性。

HP-UX 已经允许无权限的用户或进程（例如，文件所有者）在“需要了解”的前提下允许或拒绝其他用户对文件和其他对象的访问，就像通过它们的用户和（或）组的标识决定一样（请参阅 *passwd*(4) 和 *group*(4) ）。要实现此级别的控制，可以通过设置或操作文件的权限位来授权或限制所有者、组和其他用户的访问（请参阅 *chmod*(2)）。

ACL 提供比权限位更强的可选择性。 ACL 允许文件所有者或超级用户同意或拒绝一组用户和组的访问，而不是只对文件所有者和所属组进行授权。

ACL 仅可针对文件用作 UNIX 操作系统上的自由访问控制 (DAC) 机制的超集，而对于诸如进程间通信 (IPC) 对象等其他对象来说，则不支持此功能。

此联机帮助页描述的 ACL 仅能在 JFS 文件系统上实现。请参阅 *acl*(5) 中描述的 HFS 文件系统上的 ACL。

定义

因为数据访问控制是计算机安全的关键部分，我们根据《Department of Defense Trusted Computer System Evaluation Criteria》提供下列定义，以便更好地解释访问控制的概念以及它与 HP-UX 安全特性的联系：

访问　“一个主体与一个对象特定类型的交互使得信息由一端流向另一端。”主体包括“人、进程或能让信息在对象间流动或改变系统状态的设备”。对象包括文件（普通文件、目录、特殊文件以及 FIFO 等）和进程间通信 (IPC) 功能（共享内存、消息队列、信号量、套接字）。

访问控制列表 *(ACL)*
　一个访问控制列表是与指定所有可能的用户 ID 和（或）组 ID 的访问权限的文件关联的 (*user*|*group, mode*) 条目集合。

访问控制列表 *(ACL)* 条目
　ACL 中的每一个条目都指定了文件的所有者、所属组、组类、附加用户、附加组和其他用户的访问权限。

更改权限　更改 DAC 信息（权限位或 ACL 条目）的权限。将更改权限授予对象（文件）所有者和具有权限的用户。

自由访问控制 *(DAC)*
　“基于主体和（或）其从属的组的标识，对对象的访问权限进行限制的一种方法。这种控制的自由性体现在一个具有某种访问权限的主体能够将其权限（可能是间接的）传递给其他主体。”

模式　每个 ACL 条目中有三个位，它们分别代表读、写和执行/搜索权限。除了与文件系统的每一个文件相关联的 16 模式位以外，这些位也可能存在（请参阅 *glossary*(9) ）。

权限　忽略访问限制以及更改限制的功能，这些限制是由安全策略所强加的，并在一种访问控制机制中实现。在 HP-UX 中，仅超级用户和特定组的成员（请参阅 *privgrp*(4) ）是特权用户。

限制的与许可的　根据相关环境可以判断单个 ACL 条目是许可条目还是限制条目。限制条目将拒绝用户和（或）组的访问，否则，根据常规控制或可选 ACL 条目（请参阅下文），他们可能会具有该访问权限。许可条目为用户和（或）组授权，否则，根据常规控制或可选 ACL 条目，他们可能会被拒绝访问。

访问控制列表条目

访问控制列表 (ACL) 是由与指定访问权限的文件关联的单行条目的集合组成的。每个条目为一个用户 ID 或组 ID 指定一组存取权限，包括读取、写入和执行/搜索。

为了帮助理解访问控制列表和传统的文件权限之间的关系，请考虑下面的文件及其权限：

-rwxr-xr--   james   admin   datafile

文件所有者是用户 **james**。
文件的组是 **admin**。
文件名是 **datafile**。
文件所有者权限是 **rwx**。
文件组权限是 **r-x**。
文件的其他权限是 **r--**。

在 ACL 中，用户 ID 和组 ID 可由 **/etc/passwd** 中的名称或数字表示。

ACL 表示法

管理 JFS ACL 的受支持命令可识别如下的符号形式：

[**d**[**efault**]**:**]**u**[**ser**]**:**[*uid*]**:***perm*

[**d**[**efault**]**:**]**g**[**roup**]**:**[*gid*]**:***perm*

[**d**[**efault**]**:**]**c**[**lass**]**:***perm*

[**d**[**efault**]**:**]**o**[**ther**]**:***perm*

前缀是 **d:** 或 **default:** 的 ACL 条目，只可能在目录的 ACL中出现，它表示此条目的剩余部分不会用于确定对此目录的访问权限，而是应用于此目录下创建的任何文件或子目录（请参阅下面的“ACL 继承”一节）。

uid 和 *gid* 字段可包含数字用户或组 ID，或它们在 **/etc/passwd** 或 **/etc/group** 中相应的字符串。 *perm* 字段表示访问权限，既可以以符号形式表示，如 **r** 、 **w** 、 **x** 和 **-** 的组合形式，也可以以数字形式表示，如 0 到 7 八进制值表示的组合，其中 4 代表读取权限、2 代表写入权限、1 代表可执行权限。

基本 ACL 条目

创建文件时，四个基本的访问控制列表的条目将从文件的访问权限位进行映射，从而与文件的所有者、组及该文件的传统权限位相匹配。这就是所谓的“最小 ACL”。 *chmod*(2) 和 *acl*(2) 系统调用可更改基本 ACL 条目。

u::*perm* 文件所有者的基本 ACL 条目

g::*perm* 文件的组的基本 ACL 条目

c::*perm* 文件的组类的基本 ACL 条目

o::*perm* 其他的基本 ACL 条目

当 ACL 最小时，也就是说，它没有可选的 ACL 条目（请参阅下一节），这种情况下，**group** 和 **class** 的权限是等同的。

可选的 **ACL** 条目

可选的访问控制列表的条目包含附加的访问控制信息，用户可通过 *acl*(2) 系统调用设置这些信息，以更好地允许或拒绝对文件的访问。最多可指定 13 个可选 ACL 条目。

例如，下列可选的访问控制列表条目可与文件相关联：

u:mary:rwx 将读取、写入和执行权限授予给用户 **mary**。

user:george:--- 拒绝用户 **george** 的一切访问。

g:writers:rw- 将读和写权限授予给组 **writers** 的所有成员。

类条目

在一个包含了多于一个 **user** 条目和（或）多于一个 **group** 条目的 ACL 中，**class** 条目指定了可由其他 **user** 和 **group** 条目授予的最大权限。如果在 **class** 条目中未授予特定权限，则任何 ACL 条目都不得授予此权限（第一个 **user** [所有者] 条目和 **other** 条目除外）。可拒绝某一特定用户或组的任何权限。**class** 条目用作文件权限的上限。

当一个 ACL 包含了多于一个 **user** 和（或） **group** 条目时，附加的 **user** 和 **group** 条目集合将称为 **group** 类条目，因为这些附加条目的任意有效权限都受到 **class** 条目的制约。

如果 ACL 中有附加条目，则 **class** 条目将不必再与 **ls -l** 报告的所有组的权限值相等。这个特点很有用，因为它意味着 *chmod*(1) 命令可以有效地影响具有附加 ACL 条目的文件的权限。

ACL 唯一性

条目在每个 ACL 中是唯一的。只能有一种基本类型，而且任何给定的用户或组 ID 只能有一个条目。同样，只能有一种缺省基本类型，而且任何给定的用户或组 ID 只能有一个缺省条目。

ACL 继承

当一个目录的 ACL 包含缺省条目时，那些条目并不用于确定对该目录本身的访问权限的。每次在该目录下创建文件时，该目录的缺省 ACL 条目将作为非缺省 ACL 条目添加新的文件中。

例如，假设目录 **/a** 包含以下由 *getacl*(1) 报告的 ACL：

```
# file: /a
# owner: alpha
# group: uno
user::rwx
```

group::rwx
class:rwx
other:rwx
default:user:beta:r--
default:user:gamma:r--
default:group:dos:---
default:group:tres:---

于是，在 **/a** 中创建的任何新文件将使用创建者的 umask（例如 022）和目录的如下缺省 ACL 条目来初始化 ACL：

file: /a/file
owner: creator_uid
group: creator_gid
user::rw-
user:beta:r--
user:gamma:r--
group::r--
group:dos:---
group:tres:---
class:r--
other:r--

创建新的子目录时，将两次添加父目录的缺省 ACL 条目，第一次作为其非缺省 ACL 条目，而第二次作为其子目录的缺省 ACL 条目。通过这种方式，创建目录树时，缺省 ACL 将向下扩展。如果在上例中创建的是一个目录，则其 ACL 显示如下：

file: /a/dir
owner: creator_uid
group: creator_gid
user::rwx
user:beta:r--
user:gamma:r--
group::r-x
group:dos:---
group:tres:---
class:r-x
other:r-x
default:user:beta:r--
default:user:gamma:r--
default:group:dos:---

default:group:tres:---

检查访问权限算法

要分别确定一个访问进程的有效用户 ID (EUID) 和有效组 ID (EGID)，请按以下顺序进行访问权限检查：

如果此进程的 EUID 与文件所有者的相同，则授予它 **user::** 条目所指定的权限。

如果进程的 EUID 与其中一个附加 **user:***uid***:** 条目所指定的 UID 相匹配，则授予其此条目中指定的权限，该权限已与 **class** 条目指定的权限进行了按位与运算。

如果此进程的 EGID 与文件的所有组相同，则授予其 **group::** 条目所指定的权限。

如果进程的 EGID 与其中一个附加 **group:***gid***:** 条目所指定的 UID 相匹配，则授予其此条目所指定的权限，该权限已与 **class** 条目所指定的权限进行了按位与运算。

否则，授予其 **other** 条目所指定的权限。

上述检查确定访问权限后，将不执行此列表中的后续检查。

支持的 ACL 操作

通过 *acl*(2) 系统调用可对 ACL 进行设置、检索或计数。 *setacl*(1) 命令可以设置和修改 ACL， *getacl*(1) 命令可以检索 ACL。授予特定用户或组 ID 的权限可以通过 *getaccess*(1) 命令和 *getaccess*(2) 系统调用确定。具有某些 ACL 属性的文件可通过使用 *find*(1) 的 **-aclv** 选项来定位。

ACL 与 stat(2)、 chmod(2) 和 chown(2) 的交互

stat *st_mode* 字段汇总了调用者对此文件的访问权限。只有在此文件有一个或多个适用于调用者的可选条目时，它才有别于文件权限位。 *st_basemode* 字段提供了此文件实际的权限位。 *st_aclv* 字段表明此文件的 ACL 中存在可选的 ACL 条目。

st_mode 字段包含一个与用户有关的摘要，这样，使用 *stat*(2) 和 *chmod*(2) 的程序在不知道 ACL 的情况下更容易产生所需的结果，而且 *stat*(2) 可通过网络文件系统 (NFS) 提供更为合理的远程文件信息。 *st_basemode* 和 *st_aclv* 字段仅对本地文件有用。

chmod 通过 *chmod*(2) 系统调用设置组权限位可以影响文件的 **class** 条目，从而影响由附加的 **user:***uid* ： 和 **group:***gid* ： 条目所授予的权限。特别是，使用 *chmod*(2) 将文件的权限位全部设置为零将取消对文件的一切访问，而无论权限是由附加的 **user:***uid* ： 还是 **group:***gid* ： 条目所授予的。

chown 当通过 *chown*(2) 将文件的所有者或其所属组更改为具有已存在的 **user:***uid* ： 或 **group:***gid* ： 条目的 UID 或 GID 时，这些条目并不会从 ACL 中移除，但是它们将失效，因为 **user::** 或 **group::** 条目具有更高的优先级。

头文件

头文件 **<sys/acl.h>**

<sys/aclv.h> 头文件定义了以下常量，用于管理每个 ACL 中的条目数：

NACLVENTRIES 每个 ACL 的最大条目数，包括基本条目

NACLBASE 基本条目数

同时还定义了 ACL 结构 **struct acl** ，包括以下成员：

```
int a_type;     /* type of entry */
uid_t a_id;     /* group ID */
ushort a_perm;  /* see <unistd.h> */
```

<sys/aclv.h> 头文件还定义了 **a_type** 字段的有效值，以及 *acl*(2) 系统调用的 *cmd* 参数的有效值。

头文件 <sys/getaccess.h>

<sys/getaccess.h> 头文件定义与 *getaccess*(2) 结合使用的常量。

uid 的特殊参数值：

UID_EUID 使用有效的用户 ***ID***
UID_RUID 使用实际的用户 ***ID***
UID_SUID 使用保存的用户 ***ID***

ngroups 的特殊参数值：

NGROUPS_EGID 进程的有效组 ***ID***
NGROUPS_RGID 进程的实际组 ***ID***
NGROUPS_SGID 进程的保存组 ***ID***
NGROUPS_SUPP 仅适用于进程的补充组
NGROUPS_EGID_SUPP 进程的有效组 ***ID*** 加上补充组
NGROUPS_RGID_SUPP 进程的实际组 ***ID*** 加上补充组
NGROUPS_SGID_SUPP 进程的保存组 ***ID*** 加上补充组

警告

ACL 不能用于限制超级用户的访问权限。

支持的大多数（并非所有）实用程序都可以正确处理 ACL。然而，只有 *fbackup*(1M) 和 *frecover*(1M) 文件存档实用程序可以正确处理访问控制列表。在使用不能处理 ACL 而文件中又带有可选的 ACL 条目的程序时（如存档程序 *ar*(1) 、 *cpio*(1) 、 *ftio*(1) 、 *tar*(1) 和 *dump*(1M) ），请注意它们各自的参考页中包含访问控制列表的信息，以避免数据损失。

相关内容

NFS NFS 不支持远程文件的 ACL。单独的手册条目指定在这些情况下的各种系统调用、库调用和命令的行为。当在网络上传输具有可选条目的文件，或处理一个远程文件的时候，要谨慎小心，因为这些可选条目可能会以无提示方式删除。

作者

上述访问控制列表由 AT&T 开发。

文件

<sys/aclv.h> 支持 *acl*(2) 的头文件。

/etc/passwd 定义用户名、用户 ID 值和组 ID 值。

/etc/group 定义组名。

另请参阅

chmod(1)、 cp(1)、 find(1)、 getaccess(1)、 getacl(1)、 ln(1)、 ls(1)、 mv(1)、 rm(1)、 setacl(1)、 fbackup(1M)、 frecover(1M)、 fsck(1M)、 fsdb(1M)、 access(2)、 acl(2)、 chmod(2)、 chown(2)、 creat(2)、 getaccess(2)、 mknod(2)、 open(2)、 stat(2)、 aclsort(3)、 cpacl(3)、 group(4)、 passwd(4)、 privgrp(4)、 acl(5)。

名称

aio - POSIX 异步 I/O 功能

概要

#include <aio.h>

说明

POSIX 异步 I/O 功能实现 IEEE Standard 1003.1b-1993（即 Standard for Information Technology, Portable Operating System Interface (POSIX), Part 1: System Application Program Interface (API), Amendment 1: Realtime Extensions (C Language)）的第 6.7 节。它允许进程或线程启动对多个文件的同时多个读取和（或）写入操作，等待或获得请求的操作的完成通知以及检索已完成操作的状态。 POSIX 异步 I/O 功能的目的是为了允许进程或线程在 I/O 处理的同时能够进行某些计算的元素和信息处理。

接口函数

POSIX 异步 I/O 功能包括以下接口函数:

aio_read()	启动异步读取操作
aio_write()	启动异步写入操作
lio_listio()	启动异步 I/O 操作列表
aio_suspend()	等待一个或多个异步 I/O 操作的完成
aio_error()	检索异步 I/O 操作的错误状态
aio_return()	检索异步 I/O 操作的返回状态，并释放相关的系统资源
aio_cancel()	请求取消未决的异步 I/O 操作
aio_fsync()	请求同步文件的介质映像，使之与进行了异步操作后的文件保持同步

要使用这些函数，应在编译程序或链接程序命令行上指定 **-lrt** ，以链接到实时库之中。

异步 I/O 控制块

异步 I/O 控制块 (aiocb) 用作所有异步 I/O 函数的一个参数。 **aiocb** 为调用 **aio_read()** 、 **aio_write()** 或 **lio_listio()** 的异步 I/O 操作指定参数，然后，它可用作后继调用 **aio_cancel()** 、 **aio_suspend()** 、 **aio_fsync()** 、 **aio_error()** 或 **aio_return()** 的已入列的异步 I/O 操作的“句柄”。

aiocb 结构包含以下成员:

```
int      aio_fildes;              /* file descriptor */
off_t    aio_offset;              /* file offset */
void     *aio_buf;                /* location of buffer */
size_t   aio_nbytes;              /* length of transfer */
int      aio_reqprio;             /* request priority offset */
struct   sigevent aio_sigevent;   /* signal number and value */
int      aio_lio_opcode;          /* operation to be performed */
```

提供给 **aio_read()** 、 **aio_write()** 或 **lio_listio()** 的 **aiocb** 必须包含提供给正常同步的 **read()** 或 **write()** 函数调用的参数；其中， **aio_fildes** 对应于 **fildes** ， **aio_nbytes** 对应于 **nbytes** ， **aio_offset** 对应于隐含的文件地址偏移量。 **aiocb** 还可以指定一个请求优先级 delta 值 (aio_reqprio) 并发出信息，以满足唯一、实时而且异步的 I/O 要求。对于 **lio_listio()** 函数， **aio_lio_opcode** 字段指定操作是读取还是写入。

对于特定的 **aiocb** ，当一个异步 I/O 操作排入队列中后，此操作的地址将用作其他异步 I/O 函数的句柄，而且此地址仅能用于表示单个已排入队列中的操作。

aiocb 结构中定义的其他字段为将来的用途和扩展而保留。异步 I/O 功能将全部忽略这些字段。

常量声明

POSIX 标准定义的某些值是在 **aio.h** 中声明的。

函数 **aio_cancel()** 返回以下值：

AIO_CANCELED 成功取消了所有指定的异步 I/O 操作。

AIO_NOTCANCELED
至少有一个指定的异步 I/O 操作没有被成功取消。

AIO_ALLDONE 在处理请求之前，所有指定的异步 I/O 操作都已经完成。

flags 字段控制从 **lio_listio()** 函数的返回，以下值是此字段的有效取值：

LIO_WAIT 等待所有指定的操作完成。

LIO_NOWAIT 不等待操作完成就返回。

以下值是 **aio_lio_opcode** 字段中提供的操作代码，指定以 **lio_listio()** 开始的操作类型。

LIO_READ **aiocb** 指定一个异步读取操作。

LIO_WRITE **aiocb** 指定一个异步写入操作。

LIO_NOP **aiocb** 不指定任何操作，而且将以无提示方式被忽略。

操作的入列

如果检测到禁止启动操作的错误情况， **aio_read()** 和 **aio_write()** 将不入列请求。相反，它们将立即返回 **-1** ，并设置 **errno** ，以指示失败的原因。成功入列一个操作后，必须进行 **aio_error()** 调用和 **aio_return()** 调用，以确定操作的状态和错误情况，包括通常由 **read()** 和 **write()** 报告的状态和情况。请求保持已入列，并消耗进程资源和系统资源，直到调用 **aio_return()** 。

由 **lio_listio()** 入列的操作错误报告可能不如由 **aio_read()** 和 **aio_write()** 入列的操作错误报告那样及时。除资源不足外， **aio_read()** 和 **aio_write()** 的错误将立即返回 -1， **errno** 值不会导致 **lio_listio()** 停止将当前请求或后继请求入列到其列表之中。相反，部分入列操作将会成功。在这种情况下，应用程序必须使用 **aio_error()** 以确定其列表中的哪些操作已经入列，哪些操作发生了入列错误。

当以下条件之一为真时，可以说异步 I/O 操作已 完成：

- 成功执行 I/O 传输。
- 在操作的一个或多个参数中检测到错误。
- 操作被取消。

如果在用于启动操作的 **aiocb** 中指定了有效的 **sigevent** ，那么当操作完成时，将传递此信号。

异步读取和写入

异步读取和写入操作是通过使用 **aio_read()** 、 **aio_write()** 和 **lio_listio()** 接口来启动的。每个操作的参数都是在用于启动操作的 **aiocb** 中提供的。可以为 **lio_listio()** 函数调用提供一个 **aiocb** 指针列表，在这种情况下，操作的类型（读取或写入）是根据 **aiocb** 的 **aio_lio_opcode** 字段来确定的。启动后，I/O 操作可以与启动了操作的进程或线程并发执行。

利用 HP-UX 内核线程的实现，同一进程内各个相互独立的线程使用同步的 **read()** 和 **write()** 函数，使得应用程序可以实现异步 I/O 行为。但是，应用程序可能需要实现许多 POSIX 异步 I/O 功能提供的状态管理功能。

等待完成

POSIX 异步 I/O 功能支持 轮询和 通知模型。 轮询模型是通过反复调用 **aio_error()** 函数以检查操作的状态来实现的。 通知模型则是通过在 **aiocb** （用于启动操作）中指定一个 **sigevent** 来实现的。操作完成后，将发出指定的通知（如果有的话）。

aio_suspend() 函数允许应用程序等待一个或多个异步 I/O 操作的完成。必须设置 *timeout* ，使进程能够再次执行；而且，如果所期望的操作没有预期完成，进程可以采取相应的恢复操作。如果 **aio_suspend()** 引用多个操作，当任何一个操作完成时，都将返回。

检索错误

异步 I/O 操作启动之后， **aio_error()** 和 **aio_return()** 函数将检查其状态，并返回被引用的 **aiocb** 的当前状态。对于 轮询的实现， **aio_error()** 函数用于检查状态，直到发现一个完成状态；之后可以使用 **aio_return()** 来释放 **aiocb** ，使之能够被重用。

对于 通知的实现，可以确定已完成 I/O 的状态，而且可以使用单个 **aio_return()** 调用来释放 **aiocb** 。

aio_error() 和 **aio_return()** 报告的错误包括通常由 **read()** 和 **write()** 报告的所有错误以及异步 I/O 处理所特有的错误。在异步 I/O 操作启动之后但在检测到错误或操作成功完成之前， **aio_error()** 将返回 **EINPROGRESS** 。

取消

aio_cancel() 函数允许应用程序请求取消异步 I/O 操作。用于启动操作的 **aiocb** 可用作一个句柄，以标识它要求取消操作。另外，还可以请求取消文件的所有未进行的操作。并非所有异步 I/O 操作都可以被取消。

同步永久性存储

当文件或设备有多个未完成的异步 I/O 操作时， **aio_fsync()** 函数支持同步永久性存储的内容。只有指定文件的那些在函数调用时已经入列的请求才包含在同步操作之中。

文件地址偏移量

异步 I/O 操作没有固有的连续顺序。每个操作都必须指定偏移量，执行异步 I/O 操作并不会更新文件地址偏移量。

对文件设置 **O_APPEND** 标记将限制此文件的异步 I/O 值。如果设置了 **O_APPEND** ，当一个请求为下一请求提供起始偏移量之后，将顺序处理文件上的操作，并结束文件长度。尽管这有利于系统入列请求，但是请注意，不能因为大量入列必须连续处理的请求而耗尽系统资源或进程资源。

系统局限和限制

POSIX 异步 I/O 操作接口受到某些系统局限和限制。

由于每个入列的异步 I/O 操作要求为其内部控制结构分配系统内存，因此系统所允许的同时入列的异步 I/O 操作在数量上是有限的。系统上的所有活动进程能够并发入列的异步 I/O 操作的最大数量是可调的。使用带有参数 **_SC_AIO_MAX** 的 **sysconf()** 调用可以获得当前最大值。缺省最大值为 2048。除了系统级的限制，在进程级也受到限制。 **getrlimit()** 和 **setrlimit()** 系统调用使用 **RLIMIT_AIO_OPS** 参数控制这些限制。异步 I/O 操作即使已经完成，仍保持入列状态，直到为此操作调用 **aio_return()** 。

使用请求和回调机制而不使用 I/O 线程机制的异步 I/O 操作在系统级上受到物理内存量的限制，在异步 I/O 传输过程中，物理内存量可能被锁定。可以为多个 aio 请求同时锁定的系统级上的内存最大字节数是可调的。锁定的内存最大字节数可以设置为系统上可用物理内存的百分比。缺省情况下，将设置为物理内存的 10%。除了系统级的限制，在进程级上也受到限制， **getrlimit()** 和 **setrlimit()** 系统调用使用参数 **RLIMIT_AIO_MEM** 来控制对进程的限制。此外，任何原因都可能导致内存量随时被锁定，并非只有异步 I/O 才能锁定内存量，内存量锁定是由系统级的 **lockable_mem** 限定来控制的。其他系统活动，包括用 **plock()** 和（或） **mlock()** 接口进行的显式内存锁定，可能影响任何给定时刻的可锁定内存量。

能够在 **aio_reqprio** 中指定的最大优先级更改是受限的。优先级更改的最大值是可调的。使用带有参数 **_SC_AIO_PRIO_DELTA_MAX** 的 **sysconf()** 调用，可以获取当前最大值。缺省值为 20。

可以在单个 **lio_listio()** 调用中指定的异步 I/O 操作的最大数量是受限的。其限制范围是可调的。使用带有参数 **_SC_AIO_LISTIO_MAX** 的 **sysconf()** 调用可以获取当前最大值。缺省的最大值为 256。

有些异步 I/O 操作在系统级上和进程级上对同时活动的线程的数量有所限制。请参阅 *pthread*(3T) 。

编程局限和限制

异步读取操作还未完成时，如果修改与 **aiocb** （用 **aiocbp** 表示）相关联的内存的内容或者释放内存；或者对 **aiocbp->aio_buf** 表示的缓冲区进行内容修改或内存释放，即已经为 **aiocb** 调用了 **aio_return()** ，则将产生无法预见的结果。

举例

以下代码序列说明了一个异步读取操作和轮询的完成。

```
#include <fcntl.h>
#include <errno.h>
#include <aio.h>
char buf[4096];
```

```
int retval;
struct aiocb myaiocb;
bzero( &myaiocb, sizeof (struct aiocb));
myaiocb.aio_fildes = open( "/dev/null", O_RDONLY);
myaiocb.aio_offset = 0;
myaiocb.aio_buf = (void *) buf;
myaiocb.aio_nbytes = sizeof (buf);
myaiocb.aio_sigevent.sigev_notify = SIGEV_NONE;
retval = aio_read( &myaiocb );
if (retval) perror("aio:");
/* continue processing */
 ...
/* wait for completion */
while ( (retval = aio_error( &myaiocb) ) == EINPROGRESS) ;
/* free the aiocb */
retval = aio_return( &myaiocb);
```

另请参阅

aio_cancel(2)、 aio_error(2)、 aio_fsync(2)、 aio_read(2)、 aio_return(2)、 aio_suspend(2)、 aio_write(2)、 fsync(2)、 getrlimit(2)、 lio_listio(2)、 read(2)、 write(2)、 pthread(3T)。

符合的标准

aio： POSIX 实时扩展，IEEE 标准 1003.1b

名称

aio_iosize_max - lio_listio()、 aio_read() 或 aio_write() 调用中任何异步 I/O 的最大大小

值

保证安全

0

缺省值

0

允许值

0-0x10000000

建议值

0, 4096-262144

说明

对使用 *lio_listio*(2) 、 *aio_read*(2) 或 *aio_write*(2) 能够执行的异步 I/O 操作，此可调参数对其流量的大小设置限制（以字节为单位）。此参数有助于限制异步 I/O 操作可占用的内存数量。

此可调参数设置为其缺省值 0 时，不起任何作用。

当此可调参数设置为一个正数时，大小大于该数字的所有异步 I/O 都将失败，并返回 [EINVAL] 。尽管此可调参数会影响使用异步 I/O 的所有应用程序，但是它最常见的用法是与 **aio_proc_max** 一起使用。对于使用 *aio_reap*(2) 的应用程序来说，这种组合有助于限制该应用程序使用的系统资源。

更改此可调参数的人员

负责运行要求对磁盘或文件系统大量使用 AIO 的应用程序的系统管理员。

更改限制

此可调参数是动态的。对此可调参数的更改将在更改之后立即生效。系统中的所有进程（包括已经正在运行的进程），都将立即受到影响。

应该何时增加此可调参数的值

对于需要大量使用异步 I/O 的应用程序，应该增加 **aio_iosize_max** 的值。

增加此可调参数值的负面影响

将此可调参数的值从 0 增加到任何正值时，将开始强制执行此可调参数定义的限制（请参阅“说明”）。

但是，此可调参数成为正值后，进一步增加它的唯一结果是应用程序可以执行的 I/O 规模更大。设置此可调参数之后，将继续强制执行诸如 **aio_physmem_pct** 之类的限制（除非已经使用 *aio_proc_max*(5) 禁用了这些限制）。

应该何时降低此可调参数的值

当 POSIX AIO 的性能可以接受，但是要考虑带有病毒的应用程序或恶意应用程序会发出超大的 I/O 从而占用大量系统资源时，应该降低 **aio_iosize_max** 的值。当此可调参数为非零值时，它的值决不应降低到系统中运行的受信任应用程序所需的最小 I/O 大小以下。

降低此可调参数值的负面影响

只要它保持为正值，降低此可调参数的值就只会降低异步 I/O 可能有的最大大小。当此可调参数设置为 0 时，它将不再有效，且不会对各个异步 I/O 的大小实施任何限制。

更改此可调参数的同时应更改的其他可调参数

更改此参数的同时不需要更改任何其他可调参数。

但是，此可调参数最常见的用法是与 **aio_proc_max** 一起使用。这种情况下，使用 *aio_reap*(2) 的所有进程的总内存用量将限制为以下数量：

（系统中进程的最大数量） *** aio_proc_max * aio_iosize_max**

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

aio_iosize_max 由 HP 开发。

另请参阅

kctune(1M)、 sam(1M)、 gettune(2)、 settune(2)、 aio_reap(2)、 aio_read(2)、 aio_write(2)、 lio_listio(2)、 aio_physmem_pct(5)、 aio_proc_max(5)。

名称

aio_listio_max - POSIX 的最大数可在 listio() 调用中指定的异步 I/O 操作

值

无故障

256

缺省值

256

允许值

允许的最小值是 **2** 。允许的最大值是 **0x100000** 。此值被进一步限制为必须小于或等于 **aio_max_ops** 。

指定一个正整数值。

说明

如果在单个 **listio()** 调用中请求了大量 POSIX 异步 I/O 操作，则此参数将对可消耗的系统资源进行限制。当保护系统不受故障进程导致的过多异步 I/O 操作影响时，此值应该设置为足够大来满足系统编程的需要。

应该由谁来更改此可调参数？

运行需对文件系统大量使用 POSIX AIO 的应用程序的系统管理员。

对于更改的限制

此可调参数是动态的（参数的调整将在运行的系统上立即生效）。

同时还应更改哪些其他可调参数的值？

分配给 **aio_listio_max** 的值必须小于或等于 **aio_max_ops** 。

警告

所有 HP-UX 内核可调参数都是针对于特定版本的。未来的 HP-UX 版本可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

aio_listio_max 由 HP 开发。

另请参阅

kctune(1M)、 sam(1M)、 gettune(2)、 settune(2)、 aio(5)、 aio_max_ops(5)。

名称

aio_max_ops - 随时可以排队的最大 POSIX 异步 I/O 操作数

值

无故障

2048

缺省值

2048

允许值

允许的最小值是 **2** 。允许的最大值是 **0x100000** 。此值被进一步限制为必须大于或等于 **aio_listio_max** 。

指定一个正整数值。

说明

如果系统上同时有大量 POSIX 异步 I/O 操作列队，那么此参数对可消耗的系统资源进行了限制。此参数限制了争用进程的能力，使其使用大量异步 I/O 操作及所需的内存淹没系统。

每个列队的异步操作要求为其内部控制结构分配系统内存，从而使得这种限制非常必要。除了系统级的限制，还有使用参数 **RLIMIT_AIO_OPS** 对 **getrlimit()** 和 **setrlimit()** 调用进行控制的进程级的限制。

aio_listio_max 限制指定进程的单个 **listio()** 调用中可包含的操作数，以及 **aio_max_ops** 值，此值必须满足正在进行同时或几乎同时的 **listio()** 调用的所有进程的合理需求，而不会危害到整个系统的平衡。

应该由谁来更改此可调参数？

运行需对文件系统大量使用 POSIX AIO 的应用程序的系统管理员。

对于更改的限制

此可调参数是动态的（参数的调整将在运行的系统上立即生效）。

同时还应更改哪些其他可调参数的值？

分配给 **aio_max_ops** 的值必须大于或等于 **aio_listio_max** 。

警告

所有 HP-UX 内核可调参数都是针对于特定版本的。未来的 HP-UX 版本可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

aio_max_ops 由 HP 开发。

另请参阅

kctune(1M)、 sam(1M)、 gettune(2)、 settune(2)、 getrlimit(2)、 setrlimit(2)、 aio(5)、 aio_listio_max(5)。

名称

aio_monitor_run_sec - AIO 线程池执行监视的频率（以秒为单位）

值

保证安全

2

缺省值

30

允许值

1-60

建议值

1-60

说明

HP-UX 上的 POSIX AIO 实现使用内核线程对不直接支持真正的异步 I/O 的文件系统执行 I/O 操作（此区别对用户透明）。内核线程被组织到基于进程级创建的工作线程池（称为 AIO 线程池）中。由于 I/O 的线程池机制在 CPU 时间和 I/O 资源的使用上引入不同的折衷方式，有四种动态可调参数用于定制此线程池的行为: **aio_proc_threads** 、 **aio_proc_thread_pct** 、 **aio_req_per_thread** 和 **aio_monitor_run_sec** 。有关每个可调参数的详细信息，请参阅单独的联机帮助页。

可调参数 **aio_monitor_run_sec** 指定监视进程 AIO 线程池的频率。监视内容包括根据 **aio_proc_threads** 、 **aio_proc_thread_pct** 和 **aio_req_per_thread** 所指定的限制决定是否增大或缩小 AIO 线程池。

请注意，虽然 AIO 线程池可以通过自身（如发布了新的 I/O）或作为监视机制的结果增大，但监视机制是线程池缩小的主要方法。所以此可调参数有效的决定了 AIO 线程池调整自身适应给定 I/O 加载的速度。

应该由谁来更改此可调参数

运行需对文件系统大量使用 POSIX AIO 的应用程序的系统管理员。

更改限制

此可调参数为动态的。更改后的新进程启动时，对此可调参数的更改立即生效。也同样影响现有进程，但对运行中进程的扩展更改的速度取决于 **aio_monitor_run_sec** 原来的值。

应该何时增加此可调参数的值

对于具有稳定的 I/O 加载的应用程序，其 POSIX AIO 极少需要调整，应该增加 **aio_monitor_run_sec** 。另一种可能是具有突发性或定期 I/O 加载的应用程序，希望 POSIX AIO 在 I/O 活动减少的阶段中保持较大的线程池（为忙碌阶段作准备）。可以通过增加此可调参数以减少 AIO 监视更新频率来完成。

增加此可调参数值的负面影响

增加此可调参数将降低 POSIX AIO 线程池机制调整自身适应更改的 I/O 加载的速度。在应用程序首次开始发布 POSIX AIO 时会略微降低性能。

应该何时降低此可调参数的值

当应用程序 POSIX AIO 线程池调整自身适应 I/O 加载的速度需要加快时， **aio_monitor_run_sec** 的值应当降低。通常此操作会最优化性能，除了在突发 I/O 加载或有定期 I/O 加载高峰时需要较慢的适应速度。

降低此可调参数值的负面影响

POSIX AIO 线程池调节速度更快以更改 I/O 加载，因而新线程对于新 I/O 的衍生速度更快， I/O 加载减少时线程的终止也更快。除了对于突发性或定期 I/O 加载，此时需要最优化性能。

更改此可调参数的同时应更改的其他可调参数

aio_proc_threads 通过对可用于 POSIX AIO 的线程数目设置严格限制与此可调参数进行交互。

aio_proc_thread_pct 通过对可用于 POSIX_AIO 的线程数目设置严格限制与此可调参数进行交互，但此限制基于最多允许的进程线程数的百分比。允许 AIO 线程池动态响应 **max_thread_proc** 的更改。

aio_req_per_thread 定义 POSIX AIO 内核线程和服务 I/O 数目之间所需的关系。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

aio_monitor_run_sec 由 HP 开发。

另请参阅

kctune(1M) 、 sam(1M) 、 gettune(2) 、 settune(2) 、 aio_proc_threads(5) 、 aio_proc_thread_pct(5) 、 aio_req_per_thread(5)。

名称

aio_physmem_pct - 对请求回调 POSIX 异步 I/O 操作的可锁定物理内存的百分比

值

保证安全

10

缺省值

10

允许值

允许的最小值为 **5** 。允许的最大值为 **50** 。

指定一个正整数值。

说明

此参数对可由任意给定时间进行的 POSIX 异步 I/O 操作的总数锁定的系统内存的容量进行了限制。

在与某个操作相关的 **aio_return()** 调用完全终止之前（即使该操作已完成），该操作仍会保留在活动队列中，并且内存不会被释放，注意到这点很重要。

aio_physmem_pct 的值代表系统内存的百分比，并且在适当的时候，它控制的限制值可以通过联机添加或删除物理内存 (OL*) 进行调整。

使用 I/O 的请求与回调机制进行异步 I/O 操作必须锁定它们正在使用的内存。请求与回调机制只有在涉及的设备驱动器支持它时才使用。只有当 I/O 传输正在进行时才能锁定内存。

aio_physmem_pct 对可锁定的物理内存强加一个系统级的限制。通过在应用程序中使用 **setrlimit()** 系统调用，进程级可锁定内存的限制也可以自行强加（请参阅 *setrlimit*(2) ）。

更改此可调参数的人员

运行需对文件系统大量使用 POSIX AIO 的应用程序的系统管理员。

更改限制

此可调参数是动态的。对此可调参数值的任何更改不需重新引导系统即可立即生效。

可以通过 **aio_physmem_pct** 进行限制锁定的内存数量不能超过由 **unlockable_mem** 进行限制的系统级的可锁定内存的总量。

支持请求/回调机制的文件系统或设备的每次传入请求均将单独检查由此可调参数 **aio_physmem_pct** 指定的限制。当调整为较低的值时，将立即对所有新请求强制应用新限制，而且即使当前的使用情况计数较高时，此调整也将会成功。然后使用情况逐渐适应新的限制。

应该何时增加此可调参数的值

在大型服务器上，最好将 **aio_physmem_pct** 增大到更大的值（最大为 50）。

更改此可调参数的同时应更改的其他可调参数

在任何给定的时间，由于任何原因可以锁定的内存总数，（不只是由于异步 I/O，）由系统级限制 **lockable_mem** 来控制。其他系统活动，包括通过 **plock()** 和（或） **mlock()** 接口进行的显式内存锁定也会影响任意时间的可锁定内存量。

没有名为 **lockable_mem** 的内核参数，但是有一个名为 **unlockable_mem** 的参数影响它。 **lockable_mem** 的值通过从系统启动后的可用系统内存数减去 **unlockable_mem** 的值而确定。启动过程中，系统在系统控制台上显示了其可锁定内存量（连同可用内存和物理内存一起）。这些值可以在系统使用 **/sbin/dmesg** 命令运行时检索到。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

aio_physmem_pct 由 HP 开发。

另请参阅

kctune(1M)、 sam(1M)、 gettune(2)、 settune(2)、 aio(5)、 aio_return(2)、 dmesg(1M)、 mlock(2)、 plock(2)、 getrlimit(2)、 setrlimit(2)、 unlockable_mem(5)。

名称

aio_prio_delta_max - POSIX 异步 IO 请求优先级中允许的最大 delta 版（减速因子）

值

无故障

20

缺省值

20

允许值

允许的最小值是 **0** 。允许的最大值是 **20** 。

指定一个正整数值。

说明

此参数限制了 POSIX 异步 I/O 操作的优先级可以降低多少来减慢其执行。这是异步 I/O 控制块结构 **aiocb** 中允许的对 **aio_reqprio** 的最大优先级偏移值。

应该由谁来更改此可调参数?

运行需对文件系统大量使用 POSIX AIO 的应用程序的系统管理员。

对于更改的限制

此可调参数是动态的（参数的调整将在运行的系统上立即生效）。

同时还应更改哪些其他可调参数的值?

无。

警告

所有 HP-UX 内核可调参数都是针对于特定版本的。未来的 HP-UX 版本可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

aio_prio_delta_max 由 HP 开发。

另请参阅

kctune(1M)、 sam(1M)、 gettune(2)、 settune(2)、 aio(5)。

名称

aio_proc_max - 可由使用 aio_reap() 的任何进程排队的最大异步 I/O 操作数

值

保证安全

0

缺省值

0

允许值

0-0x10000000

建议值

0-0x10000000

说明

该可调参数可以对使用 *aio_reap*(2) 的进程所消耗的系统资源进行限制。在每个进程级别实施此限制，以便在 CPU 和进程数目不断增加的情况下，提高可伸缩性。

如果将该可调参数设置为 0，则它不起作用。也就是说，资源的使用将受 HP-UX 上提供的其他限制的约束（这些限制包括使用 **RLIMIT_AIO_OPS** 的 **aio_max_ops** 、 **aio_physmem_pct** 和 *setrlimit*(2) ）。使用这些限制（同时使 **aio_proc_max** 保持为 0）可确保与 POSIX 标准以及传统应用程序兼容。

但是，这些限制中的大多数需要在系统级别实施，因此，在某些情况下，它们会降低可伸缩性。在不需要与其他限制兼容的情况下，要解决此问题，可以设置 **aio_proc_max** 可调参数。

如果将 **aio_proc_max** 设置为正值，则它将成为针对使用 *aio_reap*(2) 的进程实施的唯一可调限制。如果设置了 **aio_proc_max** ，则不针对 *aio_reap*(2) 进程实施内存使用限制（例如， **aio_physmem_pct** 或 **RLIMIT_AIO_MEM** ）。但是，使用 POSIX AIO 但不使用 *aio_reap*(2) （也就是说，只使用标准的 POSIX 接口调用）的进程将继续拥有实施的所有旧限制。

对于希望获得更高的 **aio_proc_max** 可伸缩性且不放弃内存限制控制权的系统管理员来说，可以设置可调参数 **aio_iosize_max** 。该可调参数可以限制每个 I/O 的大小，并按数量有效地约束使用 *aio_reap*(2) 的所有进程占用的总内存：

（系统中进程的最大数量） *** aio_proc_max * aio_iosize_max**

使用这种方法可以全面控制系统级资源使用，而不必依赖显式系统级约束。

注释：如果设置了 **aio_proc_max** ，使用 *aio_reap*(2) 的进程仍可以使用 **RLIMIT_AIO_OPS** 限制来设置进程特定的限制。 **RLIMIT_AIO_OPS** 和 **aio_proc_max** 的最小值将是 AIO 子系统实施的值。但是，如果 **aio_proc_max** 不是零，那么与 AIO 相关的其他所有 rlimit 对 *aio_reap*(2) 用户不起作用（也就是说，不实施这些 rlimit）。

更改此可调参数的人员

负责运行要求对磁盘或文件系统大量使用AIO （使用 *aio_reap*(2) ）的应用程序的系统管理员。

更改限制

此可调参数为动态的。更改后的新进程启动时，对此可调参数的更改立即生效。但是，这些更改不影响现有的运行中的进程。（也就是说，调整时运行的任何进程将提升到“祖父级”，并遵循启动进程时此可调参数保存的值)

应该何时增加此可调参数的值

应该针对通过 *aio_reap*(2) 大量使用 AIO 的应用程序增加 **aio_proc_max** 的值。

增加此可调参数值的负面影响

如果将此可调参数从其缺省值 0 增加到正值，会发生上面所述的结果（请参阅“说明”）。但是，此可调参数使用正值后，进一步增加其值的唯一结果是更多的系统资源可用于异步 I/O。

应该何时降低此可调参数的值

当 AIO 的性能可接受，但担心会有过多的系统资源被分配给 AIO 时，应该降低 **aio_proc_max** 。

降低此可调参数值的负面影响

只要此可调参数保持正值，降低其值就会减少每个进程可以发出的 I/O 数。如果此可调参数设置为 0，则它不再起作用，系统将只实施上面所述的旧系统级可调参数（请参阅“说明”）。

更改此可调参数的同时应更改的其他可调参数

更改此参数的同时不需要更改其他可调参数。

但是，如果需要限制 AIO 的内存使用，则可以选择设置 **aio_iosize_max** 。

此外，如果 **aio_proc_max** 设置为正值，则另一种做法就是降低更旧的系统级限制值（例如 **aio_max_ops** 和 **aio_physmem_pct** ）。由于更旧的限制对 *aio_reap*(2) 用户不起作用，而 *aio_reap*(2) 用户又希望能够消耗大多数系统资源，因此这种做法非常有用（为旧限制控制的进程保留更少的资源)

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

aio_proc_max 由 HP 开发。

另请参阅

kctune(1M)、 sam(1M)、 aio_reap(2)、 gettune(2)、 settune(2)、 setrlimit(2)、 aio(5)、 aio_iosize_max(5)、aio_max_ops(5)、 aio_physmem_pct(5)。

名称

aio_proc_thread_pct - AIO 池允许的全部进程线程百分比

值

无故障

50

缺省值

70

允许值

10-100

建议值

10-90

说明

HP-UX 上的 POSIX AIO 实现使用内核线程对不直接支持真正的异步 I/O 的文件系统执行 I/O 操作。（此区别对用户透明）。内核线程被组织到进程级基础上创建的工作线程池（称为 AIO 线程池）中。由于 I/O 的线程池机制在 CPU 时间和 I/O 资源的使用上引入不同的折衷方式，有四种动态可调参数用于定制此线程池的行为：*aio_proc_threads*(5)、*aio_proc_thread_pct*(5)、*aio_req_per_thread*(5) 和 *aio_monitor_run_sec*(5)。有关这些可调参数的详细信息，请参阅单独的联机帮助页。

可调参数 **aio_proc_thread_pct** 基于进程级指定可用作 POSIX AIO 系统发布 I/O 时内核线程的线程百分比。此百分比视作 **max_thread_proc** 的百分比，其为进程可拥有线程数的上限。

此可调参数采用下列方式与 **aio_proc_threads** 进行交互：AIO 使用的最大线程数为两个可调参数定义值中的较小值，即是：

> **MIN (aio_proc_threads, aio_proc_thread_pct * max_thread_proc)**

此项允许 AIO 线程数使用 **max_thread_proc** 动态变化，但始终遵循 **aio_proc_threads** 的绝对限制。

应该由谁来更改此可调参数？

运行需要对文件系统大量使用 POSIX AIO 的应用程序的系统管理员。

对于更改的限制

此可调参数为动态的。更改后的新进程启动时，对此可调参数的更改立即生效。也同样影响现有进程，但对运行中进程的扩展更改的速度取决于可调参数 **aio_monitor_run_sec**。

何时应增加此可调参数的值？

对于自身工作不使用大量线程但需要在 POXIX AIO 子系统中获得高性能的应用程序，应当增加 **aio_proc_thread_pct**。

增加此可调参数的值会产生哪些副作用？

某些使用 POSIX AIO 同时也需要大量线程的应用程序可能发现其无法创建新线程，这是因为 POSIX AIO 线程池使用了太多进程允许的线程而结束。

此外，使用大量内核线程可能导致 CPU 的利用率增加。

何时应降低此可调参数的值?

在尝试为其他工作创建新线程时，如果 POSIX AIO 性能虽可以接受，但使用 POSIX AIO 的应用程序正在查找错误，则此时应该降低 **aio_proc_thread_pct** 。

降低此可调参数的值会产生哪些副作用?

通过最终降低可用线程数以处理 POSIX AIO 请求，可能降低 POSIX AIO 子系统的整体 I/O 处理能力。

在更改此可调参数的同时还应更改哪些其他可调参数?

aio_proc_threads 通过对可用于 POSIX_AIO 的线程数目设置严格限制与此可调参数进行交互。此操作允许强加硬性限制，而不管 **max_thread_proc** 的值是什么。

aio_req_per_thread 定义 POSIX AIO 内核线程和服务 I/O 数目之间所需的关系。

aio_monitor_run_sec 定义 AIO 线程机制遵循上述可调参数定义的限制进行自身监视的频率（以秒为单位）。

警告

所有 HP-UX 内核可调参数都特定于各个版本。未来的 HP-UX 版本可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

aio_proc_thread_pct 由 HP 开发。

另请参阅

kctune(1M) 、 sam(1M) 、 gettune(2) 、 settune(2) 、 aio_proc_threads(5) 、 aio_req_per_thread(5) 、 aio_monitor_run_sec(5)。

名称

aio_proc_threads - AIO 池允许的最大进程线程数目

值

无故障

256

缺省

1024

允许值

4-2048

建议值

32-2048

说明

HP-UX 上的 POSIX AIO 实现使用内核线程对不直接支持真正的异步 I/O 的文件系统执行 I/O 操作。（此区别对用户透明）。内核线程被组织到进程级基础上创建的工作线程池（称为 AIO 线程池）中。由于 I/O 的线程池机制在 CPU 时间和 I/O 资源的使用上引入不同的折衷方式，有四种动态可调参数用于定制此线程池的行为：*aio_proc_threads*(5) 、 *aio_proc_thread_pct*(5) 、 *aio_req_per_thread*(5) 和 *aio_monitor_run_sec*(5) 。有关这些可调参数的详细信息，请参阅单独的联机帮助页。

可调参数 **aio_proc_threads** 基于进程级指定可用作 POSIX AIO 系统发布 I/O 时内核线程的进程线程最大数目。

此可调参数采用下列方式与 **aio_proc_thread_pct** 进行交互：AIO 使用的最大线程数为两个可调参数定义值中的较小值，也就是：

MIN (aio_proc_threads, aio_proc_thread_pct * max_thread_proc)

此项允许 AIO 线程数使用 **max_thread_proc** 动态变化，但始终遵循 **aio_proc_threads** 的绝对限制。

应该由谁来更改此可调参数？

运行需要对文件系统大量使用 POSIX AIO 的应用程序的系统管理员。

对于更改的限制

此可调参数为动态的。更改后的新进程启动时，对此可调参数的更改立即生效。也同样影响现有进程，但对运行中进程的扩展更改的速度取决于可调参数 **aio_monitor_run_sec** 。

何时应增加此可调参数的值？

对于自身工作不使用大量线程但需要在 POXIX AIO 子系统中获得高性能的应用程序，应当增加 **aio_proc_threads**。

增加此可调参数的值会产生哪些副作用？

某些使用 POSIX AIO 同时也需要大量线程的应用程序会发现其无法创建新线程（这种情形可能会在当 POSIX AIO 线程池使用了太多进程允许的线程而结束时发生）。

此外，使用大量内核线程可能导致 CPU 的利用率增加。

何时应降低此可调参数的值？

在试图为其他工作创建新线程时， POSIX AIO 性能虽可以接受但使用 POSIX AIO 的应用程序发现错误，此时应该降低 **aio_proc_threads** 。

降低此可调参数的值会产生哪些副作用？

通过最终降低可用线程数以处理 POSIX AIO 请求，可能降低 POSIX AIO 子系统的整体 I/O 处理能力。

在更改此可调参数的同时还应更改哪些其他可调参数？

aio_proc_thread_pct 基于允许的进程线程最大数目的百分比为 AIO 线程数目设置限制，从而与此可调参数交互。允许 AIO 线程池动态响应 **max_thread_proc** 的更改。

aio_req_per_thread 定义 POSIX AIO 内核线程和要提供服务的 I/O 数目之间所需的关系。

aio_monitor_run_sec 定义 AIO 线程机制遵循上述可调参数定义的限制进行自身监视的频率（以秒为单位）。

警告

所有 HP-UX 内核可调参数都特定于各个发行版本。未来的 HP-UX 版本可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

aio_proc_threads 由 HP 开发。

另请参阅

kctune(1M)、 sam(1M)、 gettune(2)、 settune(2)、 aio_proc_thread_pct(5)、 aio_req_per_thread(5)、 aio_monitor_run_sec(5)。

名称

aio_req_per_thread - 未决 AIO 请求数和服务线程数之间所需比例

值

无故障

4

缺省值

1

允许值

1-100

建议值

1-100

说明

HP-UX 上的 POSIX AIO 实现使用内核线程对不直接支持真正的异步 I/O 的文件系统执行 I/O 操作。（此区别对用户透明）。内核线程被组织到进程级基础上创建的工作线程池（称为 AIO 线程池）中。由于 I/O 线程池机制在 CPU 时间和 I/O 资源的使用上引入不同的折衷方式，有四种动态可调参数用于定制此线程池的行为: *aio_proc_threads*(5) 、 *aio_proc_thread_pct*(5) 、 *aio_req_per_thread*(5) 和 *aio_monitor_run_sec*(5) 。有关这些可调参数的详细信息，请参阅单独的联机帮助页。

可调参数 **aio_req_per_thread** 基于进程级指定未决 POSIX AIO 请求数和 AIO 线程池中线程数之间所需比例。

AIO 线程池中的线程数目受可调参数 **aio_proc_thread_ptct** 和 **aio_proc_threads** 的限制，而 **aio_req_per_thread** 可调参数则确定 AIO 线程池在此限制下的行为。 **aio_req_per_thread** 通过定义每个线程负责 I/O 的数量确定未完成的 AIO 请求增加时线程池增加的大小。

应该由谁来更改此可调参数?

运行需要对文件系统大量使用 POSIX AIO 的应用程序的系统管理员。

对于更改的限制

此可调参数为动态的。更改后的新进程启动时，对此可调参数的更改立即生效。也同样影响现有进程，但对运行中进程的扩展更改的速度取决于可调参数 **aio_monitor_run_sec** 。

何时应增加此可调参数的值?

应用程序希望限制 POSIX AIO 子系统使用的线程数时，应当增加 **aio_req_per_thread** 。应用程序需要进行此操作要么是为其他工作释放出更多的进程线程，要么是为限制 POSIX AIO 内部的并发级别，也可能为了减少实际限制的 I/O 设备的加载。

增加此可调参数的值会产生哪些副作用?

通过允许更少的 POSIX AIO 请求的线程，减少并发，同时 AIO I/O 请求在等待服务时需要更长时间。这将导致增加延迟时间并在 I/O 堆栈可能要另外处理大量加载的系统上降低 POSIX AIO 性能。另一方面，每个请求的线程越少，切换的环境就越少，降低 POSIX AIO 的 CPU 利用率。

何时应降低此可调参数的值？

应用程序希望最优化并发和 POSIX AIO 请求的性能时，应该降低 **aio_req_per_thread** 。应用程序不需要大量的线程用于其他工作时应当使用此项。

降低此可调参数的值会产生哪些副作用？

降低此可调参数导致 POSIX AIO 使用更多线程处理 I/O 请求，可能增加 CPU 的使用并耗尽应用程序可能需要用于其他工作的线程。另一方面，应该提高 POSIX AIO 性能。

在更改此可调参数的同时还应更改哪些其他可调参数？

aio_proc_threads 通过对可用于 POSIX AIO 的线程的数目设置严格的限制，从而与此可调参数交互。

aio_proc_thread_pct 通过对可用于 POSIX_AIO 的线程的数目设置限制与此可调参数交互，但此限制基于允许的进程线程数的最大数的百分比。这允许 AIO 线程池动态响应 **max_thread_proc** 的更改。

aio_monitor_run_sec 定义 AIO 线程机制遵循上述可调参数定义的限制进行自身监视的频率（以秒为单位）。

警告

所有 HP-UX 内核可调参数都特定于各个版本。未来的 HP-UX 版本可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

aio_req_per_thread 由 HP 开发。

另请参阅

kctune(1M) 、 sam(1M) 、 gettune(2) 、 settune(2) 、 aio_proc_threads(5) 、 aio_proc_thread_pct(5) 、 aio_monitor_run_sec(5)。

名称

aliases - 用于 sendmail 的别名文件

概要

/etc/mail/aliases

说明

newaliases 命令（与 **sendmail -bi** 相同；请参阅 *sendmail*(1M)）从文本文件中构建 sendmail 别名数据库。缺省文本文件是 **/etc/mail/aliases** 。本地地址（本地用户名称）在别名数据库中查找并在必要时扩展，除非用户名位于反斜杠（ **** ）之后。对于给定别名，如果别名文件包含多个条目，则仅使用最后一个条目。除非 **m** 处理选项（“send to me” 选项）在 **sendmail** 命令中或在配置文件 **/etc/mail/sendmail.cf** 内已经设置，发送方不包含于任何别名扩展。例如，如果 **Joe** 发送消息到 **group** ， **group** 扩展包含 **Joe** ，消息不发送给 **joe** 。

别名文本文件的每一行格式应如下：

alias **:** *mailing-list*

邮件列表可以多行连续。每一个连续行必须以空格开始。以 # 开始的行为注释。

mailing-list 是以逗号分割的列表，包含下列一个或多个：

user-name
: 除非位于反斜杠（ **** ）之后，出现在别名扩张中的本地用户名称将自己查找别名数据库。

remote-address
: **sendmail** 识别的远程地址语法在 **sendmail** 配置文件中进行配置，通常包含 RFC-822 样式 *user*@*domain* 和 UUCP 样式 *host***!***user* 。

filename
: 此项必须为绝对路径名。 **sendmail** 仅在其驻留的目录可被读取和搜索，并且仅在文件已经存在、不可执行和可被写入时附加消息到文件。

| *command-line*
: **sendmail** 将消息作为标准输入传输到指定命令。如果 *command-line* 包含空白字符，则必须以引号（**"**）括起来。例如，

  msgs: "|/usr/bin/msgs -s"

:include: *filename*
: **sendmail** 读取接收方地址列表的 *filename* ，并将消息转发给每个地址。例如，如下别名：

  poets: ":include:/usr/local/lib/poets.list"

  读取构成组的地址列表的 **/usr/local/lib/poets.list** 。

如果在用户的主目录中存在名为 **.forward** 的文件并且归用户所有，则 **sendmail** 将为此用户重定向邮件至 **.forward** 文件中的地址列表。

.forward 或 **:include:** 文件中的地址可以是任何在别名文本文件中以 *mailing-list* 显示的地址。

sendmail 可以使用 **.forward** 文件运行程序或写入文件。由 **/etc/shells** 文件控制。如果 **.forward** 文件所有者缺少 **/etc/shells** 文件中列举的有效 Shell ，则不允许执行此类程序。用户仍可以通过将特殊字符

/SENDMAIL/ANY/SHELL/ 放置在 **/etc/shells** 文件中来执行这类程序。

别名数据库在检查接收方的 **.forward** 文件之前进行检查。别名创建完成后，主目录中存在 **.forward** 文件的本地和有效接收方会将信息转发至此文件定义的用户列表。

别名创建仅出现在本地名称。因为对任何人的消息发送都不超过一次，所以不会出现循环。

/etc/mail/aliases 定义的别名将不在 **mailx** 标头中扩展（请参阅 *mailx*(1)），但在网络和 **rmail** 标头中可见（请参阅 *mail*(1)）。

/etc/mail/aliases 仅为原始数据文件。实际别名创建信息使用 **newaliases** 以二进制格式置于文件 **/etc/mail/aliases.db** 中（请参阅 *newaliases*(1M)）。

newaliases 命令在每次 **aliases** 文件发生更改时执行以使更改生效。请注意， **ypmake** 创建的 NIS 别名映射使用 **makemap**，在 **/etc/mail** 目录中保留 **aliases.pag** 和 **aliases.dir**。

作者

aliases 由加州大学伯克利分校开发，最开始出现在 4.0BSD 中。

文件

$HOME/.forward	用户邮件转发文件
/etc/mail/aliases	别名名称原始数据文件
/etc/mail/aliases.db	别名名称数据库

另请参阅

mail(1)、 mailx(1)、 makemap(1M)、 newaliases(1M)、 sendmail(1M)。

名称

allocate_fs_swapmap - 确定何时为文件系统交换分配交换映射结构

值

无故障

0 （关闭）

缺省值

0 （关闭）

允许值

1 （打开）或 **0** （关闭）

说明

allocate_fs_swapmap 可调参数用于确定是否在文件系统交换设备初始化时分配所有需要的文件系统交换结构，或者等到需要的时候再分配。初始化时预分配所有文件系统交换结构节省了之后使用文件系统交换的时间，并防止之后由于物理内存争用而造成的分配失败。

动态分配减少了文件系统交换系统的内存占用量。

应该由谁来更改此可调参数？

任何人。

对于更改的限制

对此可调参数的更改将在下次重新引导时生效。

何时应增加此可调参数的值？

内存负载大的系统可能无法获得足够的内存来支持对文件系统交换增加的交换映射结构。虽然这些增加交换的调用可能将进行重试，但还是会失败并显示错误 [ENOMEM] 。文件系统交换增加时频繁显示 [ENOMEM] 的失败将有益于启用此可调参数。

增加此值的副作用是什么？

每个文件系统交换设备的所有交换映射结构将在内核引导进行交换初始化时进行预分配，从而增加内核的内存占用量。增大的量取决于文件系统交换设备的数量和大小。

何时应降低此可调参数的值？

内核可用内存有限且文件系统交换设备使用有限的系统应该禁用此可调参数来减少内核的内存使用。

降低此值的副作用是什么？

当内存争用很高时文件系统交换增加会失败并显示 [ENOMEM] 。

同时还应更改哪些其他可调参数的值？

无。

警告

所有 HP-UX 内核可调参数都是针对于特定版本的。未来的 HP-UX 版本可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

allocate_fs_swapmap 由 HP 开发。

名称

alwaysdump - 定义出现内核混乱时内核内存页要被转储的类别

值

保证安全

0

缺省值

0 （允许内核选择要转储的类别）。

允许值

0 到 **1024** 之间的整数值。

此整数值应当为如下所示类型的整数值的组合:

UNUSED	2 : 未使用页
USERPG	4 : 用户页
BCACHE	8 : 缓冲区缓存
KCODE	16 : 内核文本页
USTACK	32 : 进程堆栈
FSDATA	64 : 文件系统元数据
KDDATA	128 : 内核动态数据
KSDATA	256 : 内核静态数据
SUPERPG	512 : 未使用的超级页池

建议值

0 （允许内核选择要转储的类别）。

内核通常选择的值为 **480 = KSDATA + KDDATA + FSDATA + USTACK** 。

调试任何内核问题都需要 **KSDATA** 和 **KDDATA** 。调试文件系统问题时需要 **FSDATA** 。调试有关用户空间应用程序问题时需要 **USTACK** 。 **UNUSED** 、 **USERPG** 、 **BCACHE** 、 **KCODE** 和 **SUPERPG** 在调试时通常不需要。

说明

在大型系统中，出现内核混乱时转储系统内存会需要大量、甚至是不可接受的时间，具体取决于系统中安装的物理内存大小。 **dontdump** 和 **alwaysdump** 参数控制的快速转储功能提供了一种限制内核转储到特定信息类型的方法:

. 未使用的物理内存
. 用户进程
. 缓冲区缓存
. 内核代码

. 进程堆栈
. 文件系统元数据
. 内核动态数据
. 内核静态数据
. 未使用的超级页池

crashconf 命令及其关联的配置文件 **/etc/rc.config.d/crashconf** 控制出现内核混乱时内存转储中所包含的内存类别。在个别情况下，在引导进程期间，系统可能会在 *crashconf*(1M) 运行前出现混乱。此时，使用 **alwaysdump** 和 **dontdump** 可调参数可以设置配置。

存储在 **alwaysdump** 中的位映射值指定与内核混乱相关联的内存转储所包含的内存类别。

此参数的缺省值为 **0** 。此时，系统决定是否根据出现的崩溃类型进行一定类别的内存转储。

请注意，某些类型的系统崩溃要求完全崩溃转储。同时，系统运算符在进行转储时会请求完全崩溃转储。在以下任一情况下，无论用 **alwaysdump** 选择何种类别，都将执行完全转储。

更改此可调参数的人员

仅有 HP 现场工程师能更改此可调参数值。

更改限制

对此可调参数的更改将在下次重新引导时生效。要立即生效，请使用 **crashconf** 更改选择的页面。

应该何时启用此可调参数选项

出现系统崩溃时，应该启用此可调参数以包括转储中特定类别的页面。

启用此可调参数的负面影响

如果包含了分析转储不必要的页面，则转储将花更长的时间。

应该何时禁用此可调参数选项

缺省情况下将禁用此可调参数。

禁用此可调参数的负面影响

系统根据崩溃类型决定必须被转储的页面类别。

应该同时更改的其他可调参数

dontdump 可调参数不应包含与 **alwaysdump** 相同的页面类别。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

alwaysdump 由 HP 开发。

另请参阅

crashconf(1M)、 dontdump(5)。

名称

Aries - 在运行 HP-UX 的基于 Itanium(R) 的处理器系列上模拟 PA-RISC HP-UX 应用程序

说明

Aries 是一个动态的二进制转换器，它在基于 Itanium(R) 的 HP-UX 计算机上透明地模拟 32 位和 64 位 PA-RISC HP-UX 应用程序。名称 **ARIES** 可以扩展为 **A***utomatic* **R***ecompilation* 和 **I***ntegrated* **E***nvironment* **S***imulation* 。从以下方面看，Aries 是透明的：

- 没有重新编译 PA-RISC HP-UX 应用程序。
- 用户没有显式调用 Aries。

基于 Itanium(R) 的 HP-UX 操作系统内核能够识别 PA-RISC HP-UX 可执行文件，并调用 Aries 动态转换和执行应用程序。

Aries 由四个共享库组成：

/usr/lib/hpux32/pa_boot32.so

/usr/lib/hpux32/aries32.so

/usr/lib/hpux64/pa_boot64.so

/usr/lib/hpux64/aries64.so

aries32.so 共享库包含 32 位应用程序的 Aries 动态转换器。 **pa_boot32.so** 包含用于加载 **aries32.so** 的代码。同样， **aries64.so** 是 64 位应用程序的动态转换器，并且 **pa_boot64.so** 加载 **aries64.so** 。只要检测到 32 位 PA-RISC HP-UX 可执行文件，基于 Itanium(R) 的 HP-UX 内核便会调用 **pa_boot32.so** ，将 **aries32.so** 加载到内存中并启动它，以动态地转换和执行 PA-RISC HP-UX 可执行文件。同样，对于 64 位 PA-RISC HP-UX 可执行文件，则需要加载 **aries64.so** 。

Aries 由两个主要部分组成：

- 指令集体系结构 (ISA) 模拟引擎
- 环境模拟引擎

ISA 模拟引擎包含一个快速解释程序和一个动态转换器。解释程序每次模拟一个指令；当对一个 PA-RISC 基本块执行足够多的次数后，将会调用动态转换器来转换这个基本块。动态转换器生成在功能上与 PA-RISC 基本块等效的 IA64 本机代码。转换后的代码存储于转换代码缓冲区。对已转换基本块的任何进一步引用均无须进行解释或转换。

环境模拟引擎负责 PA-RISC 应用程序的系统调用、信号传递和线程管理等。控制系统控制并促进各个 Aries 子系统之间的交互，控制系统还维护 PA-RISC 应用程序的基本块地址与已转换代码缓冲区中已转换代码基本块之间的映射。

PA-RISC HP-UX GDB 调试支持

通过使用 PA-RISC HP-UX **gdb** ， Aries 支持在基于 Itanium(R) 的 HP-UX 系统上调试 PA-RISC HP-UX 应用程序。

在基于 Itanium(R) 的 HP-UX 系统上，**/usr/ccs/bin** 包含 PA-RISC **gdb32** 和 **gdb64** 二进制代码。名为 **/usr/ccs/bin/gdbpa** 的符号链接指向 **/usr/ccs/bin/gdb32** 。基于 Itanium(R) 的 HP-UX 系统的 **gdb** 调试程序可识别正在作为 PA-RISC 二进制代码调试的二进制代码，并在 Aries 下启动 **/usr/ccs/bin/gdbpa** 。

在基于 **Itanium(R)** 的系统上调试 **PA-RISC** 应用程序

要在基于 Itanium(R) 的 HP-UX 系统上使用 PA-RISC **gdb** 调试 PA-RISC 应用程序，请执行以下步骤：

1. 将环境变量 **PA_DEBUG** 设置为 1。
2. 将环境变量 **SHELL** 设置为指向 PA-RISC Shell，在执行调试的基于 Itanium(R) 的计算机上应包含 PA-RISC Shell。可以从 PA-RISC HP-UX 计算机上的 **/usr/bin** 中获得 PA-RISC Shell。
3. 将 **/usr/ccs/bin** 添加到 **PATH** 环境变量之中。
4. 按照如下方法运行 **gdb** ：

   $ gdb *PA-RISC_binary*

5. 注释：请确保用户对 **/tmp** 目录具有写入权限，并具有可创建大小为一个页面大小（由 **sysconf(_SC_PAGE_SIZE)** 系统调用获得）的临时文件的空间。

调试过程的其余部分与 PA-RISC HP-UX 平台上的调试过程相同。依《Limitations of PA-RISC GDB Support》中列出的各种限制，支持 **gdb** 调试程序的所有命令。

将 **PA-RISC GDB** 连接到 **Aries** 下的 **PA-RISC** 进程

通过连接调试程序 gdb，可以在基于 Itanium(R) 的系统上调试正在运行的 PA-RISC 进程。在将 PA-RISC gdb 连接到 Aries 下的 PA-RISC 进程之后，所有的调试操作将与在本地 PA-RISC 系统上的调试操作相同。

可以使用下列步骤将 **gdb** 连接到 Aries 下正在运行的 PA-RISC 进程：

1. 执行使用 PA-RISC gdb 调试 Aries 下的 PA-RISC 应用程序的准备步骤，如上所述。
2. **$ gdb** *PA-RISC_binary PA-RISC_processID*

调试结束后，应执行下列步骤。

1. 取消设置环境变量 **PA_DEBUG** 。
2. 恢复 **SHELL** 环境变量的初始值。

PA-RISC GDB 支持的限制

PA-RISC **gdb** 支持的当前限制如下。

1. 不支持 PA-RISC HP-UX **gdb** 之外的调试程序在基于 Itanium(R) 的系统上调试模拟的 PA-RISC 应用程序。
2. 不支持旧版本（HP-UX 10.20 和更早版本）的 **gdb** 。但是，支持使用 HP-UX 11.0（或更新版本）PA-RISC **gdb** 调试 HP-UX 10.20 应用程序。
3. 对于使用 **fork()** 和 **vfork()** 系统调用创建的子进程，PA-RISC **gdb** 的行为会有所不同。由于在 Aries 中，PA-RISC 应用程序进行的 **fork()** 调用被 **vfork()** 调用所替换，所以 PA-RISC HP-UX 平台上的 PA-RISC **gdb** 的行

为与基于 Itanium(R) 的 HP-UX 平台上在 Aries 下运行的 PA-RISC **gdb** 的行为有所不同。

4. 如果已调试的进程在系统调用中被阻塞，则将无法通过按住 **^C** 键来返回到 **gdb** 命令提示符。必须从外部终止该进程。

5. Aries 不提供真正的 MxN 线程模拟，所以不支持对与 MxN pthread 库链接的 PA-RISC 应用程序的调试。

   请注意，Aries 将与 MxN pthread 库链接的 PA-RISC 应用程序模拟为传统的 1x1 线程，使得这些应用程序仅能在 Aries 下像任何其他非 MxN 多线程应用程序一样进行调试。有关详细信，请参阅“模拟 MxN 线程”一节。

生成 **PA-RISC HP-UX** 兼容核心文件

在模拟的 PA-RISC 应用程序转储核心时， Aries 支持在基于 Itanium(R) 的系统上创建 PA-RISC 核心文件。

核心文件的大小受到 **ulimit()** 和（或） **setrlimit()** 值的限制。请参阅 *ulimit*(2) 和 *setrlimit*(2) 。

Aries 生成的 PA-RISC 应用程序核心文件名为 **core.***PA-RISC_application_name* 。

需要将 PA-RISC HP-UX 调试程序用于分析 Aries 为 PA-RISC 应用程序生成的核心文件。例如，可以在基于 Itanium(R) 的系统上使用 **/usr/ccs/bin/gdbpa** 调试由 Aries 生成的 PA-RISC 应用程序核心文件。或者，还可以将应用程序的核心文件转移到 PA-RISC 计算机上，并使用 PA-RISC 调试程序对其进行调试，尽管其过程十分繁琐而且容易出错。有关详细信息，请参阅“在不同的计算机上调试 Aries 生成的 PA-RISC 核心文件”一节。

Aries 成功写入 PA-RISC 应用程序的核心文件之后，将在 **stderr** 上输出以下消息：

ARIES32/64: Core file for PA32/64 application saved to path/core.PA_application_name

如果模拟的 PA-RISC 应用程序生成名为 **core** 或 **core.pid** 的核心文件，则它将是一个 Aries 核心转储，而不是该模拟的 PA-RISC 应用程序的核心文件。

在 PA-RISC HP-UX 计算机和基于 Itanium(R) 的 HP-UX 计算机上使用 **file** 命令可以识别由 Aries 生成的与 PA-RISC HP-UX 兼容的 **core** 文件。有关详细信息，请参阅 *file*(1) 。

HP-UX 11i v3 注释：

1. 根据 HP-UX 11i v3 中的缺省核心文件格式，Aries 以新格式生成 PA-RISC 应用程序核心文件。在新的核心文件格式中，大型 utsname 结构被写入到核心文件。有关大型 utsname 结构的详细信息，请参阅 *uname*(2) 。需要使用能够使用新格式核心文件的工具（如 **gdb** ）的最新版本。

   如果要使 Aries 生成旧格式的核心文件，则使用 Aries 的 **-core_format** 选项。请参考 “*Aries* 资源配置文件：向 *Aries* 传递选项”一节。

2. 如果已通过 **coreadm()** 系统调用或 **init** 设置更改核心文件模式和设置，则 Aries 不会遵循将核心文件写入 **core. PA-RISC_application_name** 的模式，而是按照由 **coreadm()** 系统调用或 **init** 设置指定的模式写入核心文件。

在不同的计算机上调试 **Aries** 生成的 **PA-RISC** 核心文件

在不同计算机上（PA-RISC 或基于 Itanium(R) 的系统）上调试由 Aries 生成的 PA-RISC 核心文件只能通过 PA-RISC gdb-3.0.01 或更高版本来完成。

按照下列步骤在计算机（非生成核心文件的计算机）上调试 Aries 生成的 PA-RISC 核心文件。

1. 从基于 Itanium(R) 的计算机（该计算机生成了将转移到目标计算机的核心转储文件）传输 PA-RISC 应用程序所使用的核心文件和所有共享库。

2. 将环境变量 **GDB_SHLIB_PATH** 设置为以用冒号分隔的目录路径名列表，传输过来的共享库位于该路径名所表示的目录。

3. 按照如下方法运行 **gdb**：

   $ gdb *PA_application PA_core_file*

Aries 资源配置文件：向 **Aries** 传递选项

向 Aries 传递的选项是通过资源配置文件传递的。 Aries 资源配置文件名是以下其中之一：

- **.ariesrc** 32 位 Aries 的 (**aries32.so**)
- **.aries64rc** 64 位 Aries 的 (**aries64.so**)

Aries 资源配置文件的格式

Aries 资源配置文件可能包含多个行，每个行的格式如下所示：

<full-path-of-PA-RISC-application1> <aries-options>
<full-path-of-PA-RISC-application2> <aries-options>
...
<full-path-of-PA-RISC-application*n*> <aries-options>

异常：可以指定根 (/) 路径，而不是 PA-RISC 应用程序的完整路径。如果指定了根 (/) 路径，则其后的选项将应用于正在通过 Aries 执行的所有 PA-RISC 应用程序。在第一列中指定的根目录 (/) 将作为通配符。在根 (/) 路径后指定 Aries 选项以及在与各应用程序名称匹配的行上存在 Aries 选项的情况下，所有 PA-RISC 应用程序都将在 Aries 下执行。

以下规则对 Aries 资源配置文件的格式进行了规定。

1. 应用程序路径名与 Aries 选项之间用空白字符（如制表符或空格）分隔。多个连续空白字符视为一个空白字符。

2. 应从第一列开始指定应用程序路径名和根 (/) 路径。

3. 指定无效的 Aries 选项将导致发生致命错误并终止模拟应用程序。

4. Aries 资源配置文件可以包含多个应用程序的条目。在这种情况下，应该用以各个应用程序名称开头的分隔行来进行分隔。

5. 第一列中的 **#** 字符会注释掉整行，并且 Aries 将不处理该行上的选项。

Aries 资源配置文件的搜索路径

Aries 在以下位置中搜索资源配置文件：

- 根目录 (/)
- 环境变量 **HOME** 所指向的用户主目录。

Aries 资源配置文件搜索机制具有以下特点：

1. 在根目录 (/) 中发现的 Aries 资源配置文件是系统级的 Aries 资源配置文件。该文件中指定的所有 Aries 选项将应用计算机上调用的所有 PA-RISC HP-UX 应用程序。
2. 在用户的主目录中发现的 Aries 资源配置文件是用户的专用 Aries 资源配置文件。该文件中指定的 Aries 选项应用于用户调用的所有 PA-RISC 应用程序。
3. Aries 首先处理在根目录 (/) 中发现的资源配置文件，然后处理在用户的主目录中发现的资源配置文件。
4. Aries 选项处理实质上是附加的。如果在两个目录中，即根目录 (/) 和用户的主目录中，都发现了 Aries 资源配置文件，Aries 将先处理用户的专用 Aries 资源配置文件中的选项。如果两个资源配置文件中具有相同的 Aries 选项，则用户的专用资源配置文件中指定的选项值将覆盖系统级资源配置文件中的选项值。
5. 如果 Aries 发现环境变量 **HOME** 的值为 **NULL** ，或者路径不可访访问，则将不处理用户的主目录中的 Aries 资源配置文件。
6. 如果 Aries 在根目录 (/) 或用户的主目录中都没有找到资源配置文件，则它将会使用选项的缺省值。

有用的 **Aries** 选项

大多数 PA-RISC 应用程序使用 Aries 选项的缺省值都可以正常运行。对于需要控制 Aries 的资源管理和（或）需要为其 PA-RISC 应用程序调节 Aries 的高级用户而言，以下 Aries 选项十分有用。

注释：在 Aries 资源配置文件中，对于是 **On|Off** 标记的 Aries 选项，如果有该选项，则此标记设置为 **On** ，没有相应的 Aries 选项，则此标记设置为 **off** 。不必在 Aries 资源配置文件中通过指定关键字 **On** 或 **Off** 将特定的 Aries 选项设置为 ON 或 OFF。例如，如果指定 **-order** ，则将 Aries 中的 **-order** 选项设置为 **On** 。同样，如果未指定 **-order** 选项，则将 Aries 中的 **-order** 选项设置为 **Off** 。

-ap_heap_ssz *size*

指定用于 Aries 专用堆的内存区域的最大大小（以 KB 为单位）。该区域是 Aries 堆的一部分， Aries 堆的大小由 Aries 选项 **-heap_ssz** 确定。该区域从 Aries 堆分配，并用于 Aries 的 **malloc()** 调用。 Aries 堆的其余区域用于分配新的线程。

值	**32 位 Aries**	**64 位 Aries**
缺省值：	4096 KB (4 MB)	8192 KB (8 MB)

最大值:	heap_ssz - 1024 KB	heap_ssz - 2048 KB
最小值:	1024 KB (1 MB)	2048 KB (2 MB)

如果发生 Aries 错误 **out of memory** 或 **malloc() failed** ，则应增加该值。

-ccsz *size* 设置 Aries 转换代码缓冲区所使用的内存区域的最大大小（以 KB 为单位）。

值	**32** 位 **Aries**	**64** 位 **Aries**
缺省值:	16384 KB (16 MB)	16384 KB (16 MB)
最大值:	16384 KB (16 MB)	16384 KB (16 MB)
最小值:	1024 KB (1 MB)	1024 KB (1 MB)

注释：建议将该参数值设置为缺省值或最大值。如果将该参数设置为较小的值，则将导致 Aries 转换代码缓冲区因为缓冲区满而更为频繁地刷新缓冲区。这可能导致 PA-RISC 应用程序的性能降低。

在 Aries 的未来版本中，该参数的最大值将可能大于 16 MB。

尽管 64 位 Aries 只允许该参数的最大值为 16 MB，但其实际上保留了 64 MB 的空间。当前未使用的 48 MB 空间是为以后更大的转换代码缓冲区而预留的。

-core_format *version_specifier*

指定 Aries 生成的核心文件的核心文件格式。缺省情况下，Aries 生成具有大型 **utsname** 结构的核心文件。如果要使 Aries 以非缺省的核心文件格式生成 PA-RISC 应用程序核心文件，则应使用该选项。

该参数的有效值为：

v1 （旧的核心文件格式），

v2 （新的核心文件格式）

该参数的缺省值为 **v2** 。

-corepid 导致将进程 ID（请参阅 *getpid*(2) ）追加给 Aries 所写入的 PA-RISC 应用程序的核心文件的文件名。

如果 打开该选项，则核心文件名将为 **core.***PA-RISC_application_name.PID* 。

该参数的缺省值为 **Off** 。

-heap_ssz *size*

指定 Aries 堆的最大大小（以 KB 为单位）。 Aries 堆用于分配内部的 Aries 数据结构，并用于为模拟 PA-RISC 应用程序的线程创建 Aries 线程。

Aries 堆分为两部分，第一部分用作 Aries 专用堆，其大小由 Aries 参数 **-ap_heap_ssz** 确定。Aries 堆的其余区域用于为新线程分配内存。

值	**32 位 Aries**	**64 位 Aries**
缺省值	22528 KB (22 MB)	131072 KB (128 MB)
最大值	22528 KB (22 MB)	131072 KB (128 MB)
最小值	8192 KB (8 MB)	8192 KB (8 MB)

上面列出了分别属于 32 位 Aries 和 64 位 Aries 的内核可调参数 **pa_maxssiz_32bit** 和 **pa_maxssiz_64bit** 的各个值以及其缺省值。

随着分别属于 32 位 Aries 和 64 位 Aries 的内核可调参数 **pa_maxssiz_32bit** 和 **pa_maxssiz_64bit** 值的增大， Aries 堆的最大值计算方法如下所示：

- **32 位 Aries** heap_ssz = pa_maxssiz_32bit - ssz - ccsz - 10 MB
- **64 位 Aries** heap_ssz = pa_maxssiz_64bit - ssz - ccsz - 33 MB

-issz *size* 为 PA-RISC 应用程序指定初始分配的堆栈大小（以 KB 为单位）。

值	**32 位 Aries**	**64 位 Aries**
缺省值	64 KB	64 KB
最大值	ssz 值	ssz 值
最小值	64 KB	64 KB

在启动时，Aries 通过累加参数字符串、环境字符串和 keybits 结构的大小计算 **-issz** 的最小值。如果计算得出的值大于 64 Kb，则 Aries 将使用计算得出的值设置 **-issz** 的最小值。

-noasync_chk

使 Aries 不在转换的代码中生成异步事件检查存根。该选项仅用于非多线程应用程序。该选项可为 CPU 密集型工作负荷提供更好的性能。

该参数的缺省值为 **Off** 。

-nocompat_core

使 Aries 在本地基于 Itanium(R) 的系统上生成具有 OS 字段（包含 HP-UX 的发行版本信息）的核心文件。此类核心文件不能与早期 HP-UX 平台的 **gdb** 协同工作。

该参数的缺省值为 **Off** 。

-noopt_ldcw

使 Aries 不对 PA-RISC 操作码 LDCW 和 LDCD 的转换进行优化。仅在多线程应用程序在 Aries 下不能正常运行时使用该选项。使用该选项会导致性能下降 3-4 倍。

该参数的缺省值为 **Off** 。

-nosched 使 Aries 不对转换的代码进行优化及调度。此选项仅用于当应用程序在转换的代码中运行失败时查找问题。使用该选项会导致性能下降大约 20%。

该参数的缺省值为 **Off** 。

-notrans 指示 Aries 不转换任何 PA-RISC 基本块。相反，将通过 Aries 解释程序模拟所有基本块。该选项将导致性能下降。此选项用于当 PA-RISC 应用程序使用 Aries 动态转换器运行失败时查找问题。

该参数的缺省值为 **Off** 。

-nounsafetrans

不动态转换自修改代码。该选项仅用于查找在 Aries 下运行 PA-RISC JVM 时遇到的问题。该选项会导致基于 Java 的应用程序的性能下降大约 2 倍。

该参数的缺省值为 **Off** 。

-order 指定 PA-RISC 应用程序需要对强内存进行排序。仅当需要强内存排序的 PA-RISC 应用程序在 Aries 下失败时，才需要使用该选项。此类应用程序应该是不使用内存排序语义（如 **load/store** 指令中的 **,O** 填充）但仍应对强内存进行排序的应用程序。使用本选项将导致性能下降 3-4 倍。

该参数的缺省值为 **Off** 。

-osinc *size*

为 PA-RISC 应用程序堆栈指定 Aries 中的递增内存分配的组块大小（以 KB 为单位），指定 Aries 堆等。

值	**32 位 Aries**	**64 位 Aries**
缺省值	64 KB	64 KB
最大值	-NA-	-NA-
最小值	4 KB	4 KB

在 Aries 中并不检查该参数的最大值和最小值。该参数的最大值应为以下值中的最小值：**ssz**、 **ccsz** 、 **heap_ssz** 值和 5 MB。该参数的最小值应为缺省的页面大小，即 4 KB。

注释：为该参数指定一个较小的值将可能产生过多的小内存映射区域，从而导致 PA-RISC 应用程序性能降低。这可能导致整个系统性能降低。这是因为操作系统为每个内存映射区域分配了一个保护 ID。过多的保护 ID 可能导致更多的 DTLB 丢失。

-pa_os_cpu

影响 **uname()** 和 **sysconf()** 系统调用模拟，以返回特定于 PA-RISC 处理器和 HP-UX 发行版本的信息。

即使未设置该选项，大多数 PA-RISC 应用程序也可以在 Aries 下按照期望的那样运行。当且仅当 PA-RISC 应用程序希望 **uname()** 和 **sysconf()** 系统调用返回值时，才需要使用该选项，使返回的值更为具体。如果应用程序在 PA-RISC HP-UX 系统上运行，则可以获得这些值。

该参数的缺省值为 **Off** 。

-ssz *size* 为 PA-RISC 应用程序指定最大堆栈大小（以 KB 为单位）。

值	**32 位 Aries**	**64 位 Aries**
缺省值	8192 KB (8 MB)	262144 KB (256 MB)
最大值	32768 KB (32 MB)	262144 KB (256 MB)
最小值	256 KB	256 KB

上面列出了各个值以及分别属于 32 位 Aries 和 64 位 Aries 的内核可调参数 **pa_maxssiz_32bit** 和 **pa_maxssiz_64bit** 的缺省值。

随着内核可调参数 **pa_maxssiz_32bit** 和 **pa_maxssiz_64bit** 的值逐渐增大， **-ssz** Aries 参数的最大值将与分别属于 32 位进程和 64 位进程的内核可调参数 **maxssiz** 和 **maxssiz_64bit** 的最大值相同。

Aries 使用 **getrlimit (RLIMIT_STACK, ...)** 系统调用返回的大小值或从父进程继承的值来设置 PA-RISC 应用程序堆栈大小。使用本选项可以覆盖 PA-RISC 应用程序的堆栈大小。

-ts *n* 指定 Aries 转换阈值 *n* ，其中， *n* 是一个整数值。该选项指定 Aries 在考虑对基本块进行动态转换之前应对该基本块进行解释的次数。该选项可用于在 Aries 下调节 PA-RISC 应用程序的性能。

该参数的缺省值为 **16** 。

举例：**Aries** 资源配置文件

要通过使用 32 MB PA-RISC 模拟堆栈的 Aries 来执行 32 位 PA-RISC 应用程序 **hello_world** ，可以将一个包含以下行的 **.ariesrc** 文件放置到用户的主目录或根 (/) 目录中。

/user/foo/hello_world -ssz 32768

另外，如果指定了以下行， 32 MB 的堆栈大小将应用于特定用户或所有用户调用的所有 32 位 PA-RISC 应用程序，这取决于 Aries 资源配置文件位于用户的主目录还是位于根目录。

/ -ssz 32768

下面显示的是 Aries 资源配置文件的示例。可以修改这些文件，以适合应用程序可执行路径和所需 Aries 选项。

用户专用的 Aries 资源配置文件的示例，该文件位于该用户的如下主目录中。

```
/home/user1/bin/app1       -ssz 24576 -heap_ssz 32768
/home/user1/app2           -core_format v1 -corepid
/home/user1/dbg/app3       -order
/home/user1/dll/bin/app4   -osinc 256 -issz 1024 -ts 10
/home/user1/app5           -pa_os_cpu
/home/user1/bin/app6       -ts 8 -ssz 16536 -core_format v1 -heap_ssz 24576 -osinc 1024
/usr/bin/X11/xterm         -ssz 24576 -heap_ssz 32768 -ap_heap_ssz 10240
```

/usr/local/bin/vim	***-osinc 256 -issz 3072***

系统范围内全局 Aries 资源配置文件的示例，该文件位于如下根目录中 (/) 。

/usr/bin/app1	***-osinc 256 -issz 1024 -ts 10***
/usr/local/bin/app2	***-core_format v1 -corepid***
/opt/App/bin/app3	***-order***
/usr/local/lib/app/bin/x	***-ssz 24576 -heap_ssz 32768 -ap_heap_ssz 5120***
/home/user2/app5	***-pa_os_cpu***
/home/user4/bin/app6	***-ts 8 -ssz 16536 -corefile_v1 -heap_ssz 24576 -osinc 1024***

ARIES 内存管理

Aries 受管区域 (AMA)

Aries 占用 PA-RISC 应用程序专用数据段末尾后的少量内存。这个区域称为 Aries 受管区域 (AMA)。在基于 Itanium(R) 的 HP-UX 操作系统启动 Aries 以模拟 PA-RISC 应用程序时，操作系统将专门为使用 Aries 而保留 AMA。 AMA 的大小由以下内核可调参数确定：

- **pa_maxssiz_32bit** 针对 32 位 Aries
- **pa_maxssiz_64bit** 针对 64 位 Aries

有关详细信息，请参阅 *pa_maxssiz*(5) 。

基于 Itanium(R) 的 HP-UX 操作系统将一个 **load_info** 指针传递给 Aries，该指针指向一个在 **/usr/include/crt0.h** 中定义的 **load_info_t** 类型的结构。 AMA 的开头用 **load_info->li_priv_mmf_start** 标记，AMA 的结尾用 **load_info->li_bstore_start** 标记。Aries 使用 AMA 存储以下数据：

1. Aries 的已初始化数据、未初始化数据 (BSS) 和线程本地存储(TLS)
2. 由 Aries 代码使用的 **malloc()** 调用的 Aries 堆，以及为PA-RISC 应用程序的线程模拟而创建的 Aries 线程的 Aries 堆
3. PA-RISC 应用程序的堆栈
4. 转换代码缓冲区（代码缓存区）
5. Aries 的专用数据结构
6. Aries 动态转换器用作堆的内存区域

整个 AMA 空间并不是一次分配完成的。Aries 动态地为 AMA 分配内存。由于 AMA 的存在，与 PA-RISC 本机系统相比，在 Aries 下运行的 PA-RISC 应用程序将占用更大的内存（也称为驻留内存）。驻留内存大小的增加并不等同于 AMA 大小的增加。相反，从 AMA 实际获得的内存分配量将决定驻留内存大小的增量。

pa_maxssiz_32 位 | 64 位和 Aries 参数之间的关系

32 位 Aries

进程启动时，Aries 在 AMA 中为 PA-RISC 应用程序堆栈保留空间。 PA-RISC 应用程序堆栈的最大大小的计算方法如下：

max_pa_stack_size = 以下两个因子中的最小值:

1. *ti_current* 值，该值在结构上与 *tuneinfo2_t* 类型的值相同，由使用 **maxssiz** 参数的 **tuneinfo2** 系统调用在 **/usr/include/sys/dyntune.h** 中定义。

2. Aries 根据内核可调参数 **pa_maxssiz_32bit** 的当前值而设定的一个值，如下所示:

```
if (pa_maxssiz_32bit < 128 MB)
   max_pa_stack_size = pa_maxssiz_32bit - 48 MB
else
   max_pa_stack_size = pa_maxssiz_32bit - 64 MB
```

将 AMA 分为不同的内存区域，如下所示:

```
pa_maxssiz_32bit = max_pa_stack_size +
            PA-RISC 应用程序堆栈大小 (ssz) +
            Aries 堆大小 (heap_ssz) +
            Aries 转换代码缓冲区大小 (ccsz) +
            5 MB （Aries 动态转换器堆） +
            5 MB （Aries 数据等）
```

举例:

1. 如果 32 位 PA-RISC 应用程序需求 128 MB 的堆栈大小，则将内核可调参数 **pa_maxssiz_32bit** 设置为:

   128 MB (ssz) + 22 MB (heap_ssz) + 16 MB (ccsz) + 10 MB = 176 MB

2. 如果 32 位 PA-RISC 应用程序要求堆栈大小的最大可能值，则将内核可调参数 **pa_maxssiz_32bit** 设置为:

   383 MB (ssz) + 22 MB (heap_ssz) + 16 MB (ccsz) + 10 MB = 431 MB

3. 如果 32 位 PA-RISC 应用程序要求 128 MB 的 Aries 堆，假定应用程序堆栈为 32 MB，则将内核可调参数 **pa_maxssiz_32bit** 设置为:

   32 MB (ssz) + 128 MB (heap_ssz) + 16 MB (ccsz) + 10 MB = 186 MB

64 位 Aries

进程启动时，Aries 在 AMA 中为 PA-RISC 应用程序堆栈保留空间。 PA-RISC 应用程序堆栈的最大大小的计算方法如下所示:

max_pa_stack_size = 以下两个因子中的最小值:

1. *ti_current* 值，该值在结构上与 **tuneinfo2_t** 类型的值相同，由使用 **maxssiz_64bit** 参数的 **tuneinfo2** 系统调用在 **/usr/include/sys/dyntune.h** 中定义。

2. max_pa_stack_size = pa_maxssiz_64bit - 224 MB

将 AMA 分为不同的内存区域，如下所示:

pa_maxssiz_64bit = max_pa_stack_size +
PA-RISC 应用程序堆栈大小 (ssz) +
Aries 堆大小 (heap_ssz) +
Aries 转换代码缓冲区大小 (ccsz) +
8 MB （Aries 动态转换器堆） +
25 MB （Aries 数据等）

举例:

1. 如果 64 位 PA-RISC 应用程序要求 512 MB 的堆栈大小，则将内核可调参数 **pa_maxssiz_64bit** 设置为:

   512 MB (ssz) + 128 MB (heap_ssz) + 64 MB (ccsz) + 33 MB = 737 MB

2. 如果 64 位 PA-RISC 应用程序要求堆栈大小的最大可能值，则将内核可调参数 **pa_maxssiz_64bit** 设置为:

   1024 MB (ssz) + 128 MB (heap_ssz) + 64 MB (ccsz) + 33 MB=1249 MB

3. 如果 64 位 PA-RISC 应用程序要求 512 MB 的 Aries 堆，并假定应用程序堆栈为 256 MB，则将内核可调参数 **pa_maxssiz_64bit** 设置为:

   256 MB (ssz) + 512 MB (heap_ssz) + 64 MB (ccsz) + 33 MB = 865 MB

Aries 下的 **PA-RISC** 应用程序线程模拟

计算所要求的 **Aries** 堆大小

要模拟 PA-RISC 应用程序的线程，无论 PA-RISC 应用程序何时创建新线程，Aries 都需要创建自己的本机线程。要创建新线程，Aries 需要分配内存，以容纳新线程的线程专用数据、线程堆栈和线程后备存储区域。对在 Aries 中创建线程的内存要求如下所示:

- 对于 32 位 Aries： 215 KB
- 对于 64 位 Aries： 280 KB

除用于分配新线程所需的内存外，Aries 还需要一些用于其内部动态数据结构分配的内存。 Aries 专用堆的大小由 Aries 参数 **-heap_ssz** 确定。Aries 将堆区域分为两部分 — 第一部分用于 Aries 专用堆，第二部分用于分配新线程。因此，Aries 所需的空间量计算如下:

所要求的 Aries 堆大小 =
-ap_heap_ssz KB + （PA-RISC 应用程序的线程数量） *
（分配一个 Aries 线程所要求的内存，以 KB 为单位）

Aries 可以创建的线程的最大数量

当内核可调参数 **pa_maxssiz_32bit** 和 32 位 Aries 参数都为缺省值的情况下，PA-RISC 应用程序最多可以在 32 位 Aries 下创建 86 个线程。当内核可调参数 **pa_maxssiz_64bit** 和 64 位 Aries 参数都为缺省值时，同样的应用程序在 64 位 Aries 下最多可以创建 439 个线程。

请检查内核可调参数 **max_thread_proc** 的值，确保它不小于 PA-RISC 应用程序需要创建的线程数量。

如果 PA-RISC 应用程序所要创建的线程数量大于使用 Aries 参数和内核可调参数 **pa_maxssiz_32|64bit** 的缺省值所

能创建的最大线程数量，则通过在 Aries 资源配置文件中指定 **-heap_ssz** 选项来增加 Aries 堆的大小。

注释：减少 PA-RISC 应用程序的堆栈大小（Aries 选项 **-ssz** ）将在 AMA 中产生更多的可用内存，但是 Aries 不能自动使用这些内存为 Aries 堆分配更多空间。要改变 Aries 堆的大小，必须在 Aries 资源配置文件中设置 Aries 选项 **-heap_ssz** 。

模拟 **MxN** 线程

Aries 支持与 MxN pthread 库链接的 PA-RISC 应用程序。在该版本的 HP-UX 中，PA-RISC MxN pthread 库是在基于 Itanium(R) 的系统上发布的。

Aries 的将来版本可能支持真正的 MxN pthread 模拟。在此之前，Aries 支持以传统的 1x1 模式与 MxN pthread 库链接的 PA-RISC HP-UX 应用程序的模拟。 Aries 通过在内部将环境变量 **PTHREAD_COMPAT_MODE** 设置为 **1** 来实现这一功能。

在 **Aries** 下模拟 **PA-RISC** 应用程序堆栈

基于 Itanium(R) 的 HP-UX 内核所分配的堆栈被 Aries 用作自己的本机堆栈。 Arie 在 AMA 中为 PA-RISC 应用程序分配堆栈。大多数 PA-RISC 应用程序都可以在 Aries 设置的缺省堆栈大小下正常运行。但是，如果 PA-RISC 应用程序发生核心转储失败，并提示以下出错信息，则应在 Aries 资源配置文件中使用 Aries 选项 **-ssz** 来增加 PA-RISC 应用程序堆栈大小。

```
ARIES32|64 Limitation/Error
PID xxxxx received SIGSEGV for stack growth failure

Possible causes - insufficient memory or swap space,
  or stack size exceeded pa_maxssiz_32|64bit

Aries does not support applications which are nearly or completely
maxed out on their data segment address space usage. This is because
Aries consumes small amount of virtual memory address space of
application

Terminating emulation
```

Aries 如何设置 **PA-RISC** 应用程序堆栈大小

进程启动时，Aries 根据以下情况设置 PA-RISC 应用程序堆栈大小：

1. 如果 Aries 资源配置文件中有 Aries 选项 **-ssz** ， Aries 将首先检查该选项。然后，Aries 用 **-ssz** 的值设置 PA-RISC 应用程序堆栈大小的值。

2. 如果没有 Aries 选项 **-ssz** ，则 Aries 将检查父进程是否指定了对堆栈大小的限制。如果指定了限制，则 Aries 将 PA-RISC 应用程序堆栈的大小设置为父进程指定的大小。父进程可以使用系统调用 **setrlimit (RLIMIT_STACK, ...)** 修改堆栈大小限制。

3. 如果没有 Aries 选项 **-ssz** ，而且父进程没有限制堆栈大小，则 Aries 将使用系统调用 **getrlimit (RLIMIT_STACK, ...)** 获得的值来设置 PA-RISC 应用程序堆栈大小。如果应用程序是在 Shell 中调用的，则该值与用户在 Shell 中使用 **ulimit -s** 设置的值相同。

有关 Aries 可以分配的堆栈大小最大可能值的详细信息，请参阅“pa_maxssiz_32 位 | 64 位和 Aries 参数之间的关系”一节。

注释：基于 Itanium(R) 的系统的本地进程（32 位和 64 位）与模拟的 PA-RISC 进程（32 位和 64 位）之间可以继承堆栈大小。它受到可用堆栈空间的限制。例如，64 位进程可以设置一个在 32 位进程的地址空间中所不能分配的堆栈大小。在这种情况下，32 位进程将使用最大的堆栈空间。

Aries 性能

出于性能比较的目的，假定对 PA-RISC 和基于 Itanium(R)2 的系统的进行相同的计价和完整配置。由于处理器缓存、内存插槽和总线速度的不同， PA-RISC 和基于 Itanium(R)2 的系统不可能在机器资源方面完全相同。

在基于 Itanium(R)2 的系统（运行速度为 1600 MHz）的 Aries 下运行的 PA-RISC 应用程序的相对性能如下表所示。

应用程序类别	比较对象 **Itanium(R)2 1600 MHz** 系统	比较对象 **PA8700 750 MHz** 系统	比较对象 **PA8800 900 MHz** 系统
常规（I/O、内存和系统密集型操作的组合）	60-70%	90-100%	80-90%
整数密集型	40-45%	90-100%	65-70%
浮点密集型	15-20%	40-50%	30-35%
基于 Java 应用程序	40-65%	70-80%	60-75%
多线程应用程序	40-50%	60-70%	50-60%

上表所示的性能数字仅为指示性的。应用程序的实际性能可能会随应用程序的执行配置文件的不同而变化。建议在进行实际部署之前，检查基于 Itanium(R)2 的系统中 Aries 下的应用程序的性能。

Aries 性能具有以下的特点：

1. 常规应用程序具有 I/O 密集型、内存密集型和系统密集型操作的均衡组合。此类应用程序在 Aries 下运行时具有良好的性能。

2. 整数密集型应用程序在算法计算方面消耗了很多时间。此类应用程序在 Aries 下运行时具有良好的性能。

3. 浮点密集型应用程序通常为科学模拟、建模应用程序。此类应用程序在 Aries 下运行时性能会受到影响，这是由于 PA-RISC 和 Itanium(R)2 处理器的浮点操作的功能和资源在体系结构上存在差异造成的。

4. 除非基于 Java 的应用程序使用 JNI 代码，否则，建议使用 Itanium(R)2 本地 JVM 运行应用程序。Aries 下基于 Java 的应用程序的性能差异很大，具体情况取决于执行配置文件。

5. 多线程 PA-RISC 应用程序的 Aries 性能低于同一应用程序的非线程部分的性能，多线程应用程序会导致 Aries 中线程的同步开销。仅当线程在 Aries 中到达一个挂起安全点时，Aries 才可以将其挂起以便为应用程序提供正确的模拟环境。

6. 如果 PA-RISC 应用程序使用性能函数库，如 **HP mlib** ，则不适合在 Aries 下运行。性能函数库（例如 **HP mlib** ）内核是用汇编语言手动编写的，而且针对缓存行为和指令资源进行调节，使性能接近计算机上的理论最高值。像 Aries 这样的软件模拟程序不能为最大化硬件资源利用而模拟应用程序的具体行为。

7. 进行 **OpenGL** 密集运算的 PA-RISC 应用程序的性能将下降。这是因为基于 Itanium(R) 的计算机上的 **ogld** 守护进程是本地进程，模拟应用程序不能通过直接与 **ogld** 守护进程通信将图形输出发送给本机显卡。此类 PA-RISC 应用程序通过虚拟内存驱动程序 (VMD) 将图形数据发送给基于 Itanium(R) 的计算机。 VMD 是在 Aries 下模拟的。图形数据的显示过程比较缓慢。相反，此类应用程序应通过使用 GLX 协议模式来使用 **OpenGL display lists** 。使用 **OpenGL** 进行的图形数据显示过程明显快于虚拟内存驱动程序方法。

Aries 支持的应用程序

Aries 支持模拟所有的 PA-RISC HP-UX 应用程序。这意味着支持所有的 HP-UX 进程间通信机制，如信号量、共享内存和套接字等。 Aries 支持模拟 PA-RISC 应用程序与本机运行的基于 Itanium(R) 的应用程序之间的所有进程间通信。还支持 Aries 下 PA-RISC 应用程序的信号行为和异常行为。在 Aries 下，仅有很小的一个 PA-RISC 应用域的子集不受支持。要想确定一个给定的应用程序能否在 Aries 下正确运行，应查看该应用程序是否属于“ ARIES 限制情况”一节中列出的 Aries 限制情况。

Aries 限制情况

除以下限制情况（或例外情况）以外，Aries 支持模拟所有的 PA-RISC HP-UX 应用程序：

1. Aries 不支持加载基于 Itanium(R) 的共享库的 PA-RISC 应用程序。换句话说，不支持混合使用 PA-RISC 二进制代码和基于 Itanium(R) 的共享库。 Aries 只支持纯 PA-RISC 应用程序，如仅与 PA-RISC 库静态或动态链接的二进制代码。但是支持基于 Itanium(R) 的本地应用程序与模拟 PA-RISC 应用程序之间的任何类型的进程间通信。

2. Aries 的当前发行版本支持在“HP-UX 11i v3”及以下版本上运行的 PA-RISC HP-UX 应用程序。 Aries 不支持在 HP-UX 8.x 或更早版本上编译的应用程序。但是此类应用程序可以在较新版本的 HP-UX（例如“HP-UX 11i v3”及以前版本）上正常运行。

3. Aries 不支持具有权限限制的 PA-RISC 指令。因此，不支持设备驱动程序和可加载的内核模块。

4. 对于假定"计算机时间用于执行应用程序代码和（或）系统调用的特定片断"的 PA-RISC 应用程序， Aries 不能保证正确地模拟此类应用程序。从理论上讲，这类应用程序是未同步的应用程序，因此需要使用适当的同步技术，如互斥锁、信号量等等，对它们进行更正。在实际运用中，此类未同步应用程序十分罕见。

5. Aries 不支持使用 **ptrace()** 或 **profil()** 系统调用的应用程序。但是，Aries 支持使用 PA-RISC **gdb** 调试模拟应用程序。有关其他信息，请参阅"PA-RISC HP-UX GDB 调试支持"一节。该限制可能影响调试程序；调试程序通常是无法以任何方式进行移植的。

6. Aries 消耗小部分的应用程序虚拟内存地址空间。因此，Aries 不支持几乎或完全消耗了其虚拟地址空间的应用程序。在实际运用中，此类应用程序十分罕见。

7. Aries 支持 **fork()** 和 **vfork()** 系统调用。但是，Aries 不支持依赖于 **fork()** 和 **vfork()** 之间的区别的应用程序。但是程序员可以理解大多数应用使用 **vfork()** 系统调用的目的。标准的应用程序依赖于 **fork()** 和 **vfork()** 系统之间的差别是极为罕见的情况。有关详细信息，请参阅 *vfork*(2) 和 *fork*(2) 。

8. 当模拟应用程序进行返回进程相关信息的系统调用时，在模拟下，Aries 返回与 PA-RISC 2.0 进程有关的信息，即使模拟应用程序运行在基于 Itanium(R) 的计算机上。例如，使用参数 **_SC_CPU_VERSION** 的系统调用 **sysconf**(2) 返回 **CPU_PA_RISC2_0** 。这是一个 Aries 策略：对于模拟应用程序，基于 Itanium(R) 的系统上的完整 PA-RISC 环境是可见的。如果应用程序要求确定自己是否能够在基于 Itanium(R) 的计算机上运行，则可以使用 **system()** 调用和基于 Itanium(R) 的本地命令 **getconf** 获得所需要的字段。有关详细信息，请参阅 *system*(3S) *getconf*(1) 和 *sysconf*(2) 。

作者

Aries 由 HP 开发。

另请参阅

file(1) 、 gdb(1) 、 getconf(1) 、 init(1M) 、 coreadm(2) 、 execve(2) 、 fork(2) 、 getrlimit(2) 、 setrlimit(2) 、 signal(2)、 sysconf(2)、 ulimit(2)、 uname(2)、 vfork(2)、 system(3S)、 core(4)、 pa_maxssiz(5)。

名称

ascii - ASCII 字符集映射

概要

cat /usr/share/lib/pub/ascii

说明

/usr/share/lib/pub/ascii 文件提供 ASCII 字符集映射，为每个字符提供等效的八进制和十六进制形式。文件包含下列文本：

```
|000 nul|001 soh|002 stx|003 etx|004 eot|005 enq|006 ack|007 bel|
|010 bs |011 ht |012 nl |013 vt |014 np |015 cr |016 so |017 si |
|020 dle|021 dc1|022 dc2|023 dc3|024 dc4|025 nak|026 syn|027 etb|
|030 can|031 em |032 sub|033 esc|034 fs |035 gs |036 rs |037 us |
|040 sp |041  ! |042  " |043  # |044  $ |045  % |046  & |047  ' |
|050  ( |051  ) |052  * |053  + |054  , |055  - |056  . |057  / |
|060  0 |061  1 |062  2 |063  3 |064  4 |065  5 |066  6 |067  7 |
|070  8 |071  9 |072  : |073  ; |074  < |075  = |076  > |077  ? |
|100  @ |101  A |102  B |103  C |104  D |105  E |106  F |107  G |
|110  H |111  I |112  J |113  K |114  L |115  M |116  N |117  O |
|120  P |121  Q |122  R |123  S |124  T |125  U |126  V |127  W |
|130  X |131  Y |132  Z |133  [ |134  \ |135  ] |136  ^ |137  _ |
|140  ` |141  a |142  b |143  c |144  d |145  e |146  f |147  g |
|150  h |151  i |152  j |153  k |154  l |155  m |156  n |157  o |
|160  p |161  q |162  r |163  s |164  t |165  u |166  v |167  w |
|170  x |171  y |172  z |173  { |174  | |175  } |176  ~ |177 del|

| 00 nul| 01 soh| 02 stx| 03 etx| 04 eot| 05 enq| 06 ack| 07 bel|
| 08 bs | 09 ht | 0a nl | 0b vt | 0c np | 0d cr | 0e so | 0f si |
| 10 dle| 11 dc1| 12 dc2| 13 dc3| 14 dc4| 15 nak| 16 syn| 17 etb|
| 18 can| 19 em | 1a sub| 1b esc| 1c fs | 1d gs | 1e rs | 1f us |
| 20 sp | 21  ! | 22  " | 23  # | 24  $ | 25  % | 26  & | 27  ' |
| 28  ( | 29  ) | 2a  * | 2b  + | 2c  , | 2d  - | 2e  . | 2f  / |
| 30  0 | 31  1 | 32  2 | 33  3 | 34  4 | 35  5 | 36  6 | 37  7 |
| 38  8 | 39  9 | 3a  : | 3b  ; | 3c  < | 3d  = | 3e  > | 3f  ? |
| 40  @ | 41  A | 42  B | 43  C | 44  D | 45  E | 46  F | 47  G |
| 48  H | 49  I | 4a  J | 4b  K | 4c  L | 4d  M | 4e  N | 4f  O |
| 50  P | 51  Q | 52  R | 53  S | 54  T | 55  U | 56  V | 57  W |
| 58  X | 59  Y | 5a  Z | 5b  [ | 5c  \ |  5d ] | 5e  ^ | 5f  _ |
| 60  ` | 61  a | 62  b | 63  c | 64  d | 65  e | 66  f | 67  g |
| 68  h | 69  i | 6a  j | 6b  k | 6c  l | 6d  m | 6e  n | 6f  o |
```

```
| 70  p | 71  q | 72  r | 73  s | 74  t | 75  u | 76  v | 77  w |
| 78  x | 79  y | 7a  z | 7b  { | 7c  | | 7d  } | 7e  ~ | 7f del|
```

控制字符

下表使用八进制、十进制、十六进制值列举了 ASCII 控制字符集。要从键盘获得相应字符，请使用指定的组合键。

要在使用 **vi** 或 **ex** 编辑器时将控制字符放在文件中，请在键入所需控制字符之前键入 Ctrl-**v** 。

八进制	十进制	十六进制	显示	符号	字符名称	按键
000	*000*	*00*	无	NUL	Null	Ctrl-Shift-@
001	*001*	*01*	*^A*	SOH	标头开始	Ctrl-**A**
002	*002*	*02*	*^B*	STX	文本开始	Ctrl-**B**
003	*003*	*03*	*^C*	ETX	文本结束	Ctrl-**C**
004	*004*	*04*	*^D*	EOT	传输结束	Ctrl-**D**
005	*005*	*05*	*^E*	ENQ	询问	Ctrl-**E**
006	*006*	*06*	*^F*	ACK	确认	Ctrl-**F**
007	*007*	*07*	*^G*	BEL	警告	Ctrl-**G**
010	*008*	*08*	*^H*	BS	退格	Ctrl-**H**
011	*009*	*09*	*^I*	HT	横向制表符	Ctrl-**I**
012	*010*	*0A*	*^J*	LF	换行符	Ctrl-**J**
013	*011*	*0B*	*^K*	VT	纵向制表符	Ctrl-**K**
014	*012*	*0C*	*^L*	FF	换页符	Ctrl-**L**
015	*013*	*0D*	*^M*	CR	回车符	Ctrl-**M**
016	*014*	*0E*	*^N*	SO	向外切换	Ctrl-**N**
017	*015*	*0F*	*^O*	SI	向内切换	Ctrl-**O**
020	*016*	*10*	*^P*	DLE	数据链接转义符	Ctrl-**P**
021	*017*	*11*	*^Q*	DC1	设备控制 1	Ctrl-**Q**
022	*018*	*12*	*^R*	DC2	设备控制 2	Ctrl-**R**
023	*019*	*13*	*^S*	DC3	设备控制 3	Ctrl-**S**
024	*020*	*14*	*^T*	DC4	设备控制 4	Ctrl-**T**
025	*021*	*15*	*^U*	NAK	否定确认	Ctrl-**U**
026	*022*	*16*	*^V*	SYN	同步	Ctrl-**V**
027	*023*	*17*	*^W*	ETB	结束传输块	Ctrl-**W**
030	*024*	*18*	*^X*	CAN	取消	Ctrl-**X**
031	*025*	*19*	*^Y*	EM	介质结束	Ctrl-**Y**
032	*026*	*1A*	*^Z*	SUB	替换	Ctrl-**Z**
033	*027*	*1B*	*^[*	ESC	转义	Ctrl-**[**
034	*028*	*1C*	*^*	FS	文件分隔符	Ctrl-****
035	*029*	*1D*	*^]*	GS	组分隔符	Ctrl-**]**
036	*030*	*1E*	*^^*	RS	记录分隔符	Ctrl-Shift-^
037	*031*	*1F*	*^_*	US	单元分隔符	Ctrl-Shift-_
177	*127*	*7F*	*^?*	DEL	删除	DEL

警告

请注意，某些 HP-UX 子系统（例如键盘接口、Windows 系统和其他系统软件）可能使用选定的键盘控制字符用于特殊用途，此时可能导致意外的结果。

文件

/usr/share/lib/pub/ascii

已过时

名称

Audio - 通过 HP VUE 使用的音频工具（已过时）

说明

此联机帮助页介绍通过 HP VUE 可用于播放、录制和编辑声音的音频工具。这些包括工具“音频设置”、“音频安全”、“音频编辑器”、“音频控制面板”、“音频文件和数据格式”以及“音频库”。*Audio*(5) 还提供了从 HP-UX 命令行中使用其他音频工具的信息。

音频设置要求

要使用音频工具，既要访问音频客户端，又要访问服务器软件。该软件是 HP-UX 的一部分。服务器需要有音频硬件的工作站或 X 工作站。

除了 720、730 和 750，音频硬件内置于所有 700 系列计算机；可以将这些型号升级为 725、735 或 755 等具有音频硬件的型号。请注意，旧的 705（也就是具有 8MB HP-UX 的 705）不包含音频硬件。

要在 X 工作站上使用音频，需要包含音频附件工具包的 HP ENVIZEX 或 ENTRIA X 工作站。

大多数情况下，在一个系统下使用音频客户端和服务器软件。但是，如果需要在远程工作站或 X 工作站上运行音频服务器，则请参阅 *aserver*(1M) 。音频数据文件可以驻留在系统或第三方系统中。

音频安全

音频的安全性设置为仅允许本地工作站用户访问。如果需要允许远程系统访问工作站上的音频，请参阅 *asecure*(1M) 。

音频编辑器

音频编辑器是基于 OSF/Motif 的工具，具有播放、录音和编辑功能。编辑器显示波形，使编辑和播放音频片断更加容易。

用户可以打开音频文件并播放，查看其波形，使用波形控制编辑文件。请使用“音频控制面板”设置输出设备。

要录音，首先连接麦克风或系统支持的其他音频设备；可以为 CD 或磁带播放机。要完成连接，请参阅“音频编辑器任务”联机帮助（“音频编辑器任务”一节）或系统用户手册。

可以使用编辑器创建和录制音频文件。

> 要从“常规工具箱”中启动编辑器，打开“媒体工具箱”并将音频文件拖放至“音频编辑器”控制或双击控制。
>
> 要在终端窗口启动编辑器，请按如下输入：
>
> **/opt/audio/bin/audio_editor** [*pathname*]

通过编辑器的右上角的 **Help** 菜单可以使用联机帮助。

音频控制面板

“音频控制面板”是基于 OSF/Motif 的工具，用于设置音量及选择回放音频设备。

音量控制影响该工作站或 X 工作站上所有客户端系统播放的音量。“音频控制面板”还包括“停止”按钮以停止

当前播放操作。

还可以使用“音频控制面板”选择用于回放的设备（耳机、内置扬声器或连接到线性输出的设备，如外置扬声器）。该选项控制双击音频文件或使用“音频编辑器”播放文件时播放音频的设备。缺省输出设备为内部（内置）扬声器。

要从 HP VUE 中启动“音频控制面板”，请单击“HP VUE 前面板”上的“音频”控制。

要在终端窗口启动“音频控制面板”，请键入以下内容：

/opt/audio/bin/AudioCP

如果系统上有使用早期音频软件版本开发的音频应用程序，则这些应用程序可使用 **SPEAKER** 环境变量确定其输出设备。通过修改 **$HOME/.vueprofile** 文件，可以为由 HP VUE 启动的全部应用程序设置 **SPEAKER** 变量。**SPEAKER** 变量可以为外置（耳机、线性输出）或内置（内置扬声器）。

要为 POSIX 或 Korn Shell 设置 **SPEAKER** 变量，请输入：

SPEAKER=internal
export speaker

要为 C Shell 设置 **SPEAKER** 变量，请输入：

setenv SPEAKER internal

音频文件和数据格式

受支持的音频文件包含下列三种文件格式中之一的未压缩音频数据：常规、RIFF/波形 或原始数据。每个文件还需要正确的文件扩展名。对于这三种文件格式，“音频编辑器”联机帮助列举了应用的数据格式和文件扩展名。

扩展名确定了显示于“文件管理器”中相应的图标。要播放音频文件，可以拖放文件图标至“音频编辑器”或“音频控制面板”或双击文件图标。

如果要为文件添加扩展名（或转换文件格式），则建议使用 **/opt/audio/bin/convert** 命令。请参阅 *convert*(1)。但也可以重命名文件以使其能够播放。使用此文件名格式：

*filename***.***rate***.***data_type*

rate 和 *data_type* 变量可使用为 **convert** 的 **-drate** 和 **-ddata** 选项定义的值。如果需要，可以省略 *rate* 变量。使用此文件名格式：

*filename***.***data_type*

音频库

HP-UX 包括一个用于构建音频工具的“音频库”。如果已经订购并安装了 User Environment Developer’s Kit，则可以使用“音频库”创建其他音频应用程序。

HP-UX“音频库”包含 C 程序可以用来处理音频的函数。函数与音频服务器交互，使应用程序可以录制和播放音频数据文件并将音频数据文件的格式从一种格式转换为另一种格式。

有关音频程序的更多信息，请参考《Using the Audio Developer’s Kit》。

已过时

作者

音频库、音频编辑器和音频控制面板由 HP 开发。

另请参阅

attributes(1)、 convert(1)、 send_sound(1)、 asecure(1M)、 aserver(1M)。

《Using Audio Developer's Kit》

名称

audit - HP-UX 审核系统简介

说明

审核系统的目的是，记录主体访问对象的实例，同时可以对绕开保护机制及滥用权限的任何（重复的）尝试进行检测，因而起到防止滥用系统并显示系统潜在安全漏洞的作用。

用户和事件选择

审核系统为管理员提供了一种选择要审核的用户和活动的机制。

在已转换为受信任模式的系统上，管理员向用户分配称为 “审核 **ID**” 的唯一标识符，该标识符在整个用户历史记录中保持不变。请参阅有关受信任模式的“警告”一节。 **audusr** 命令用于指定那些要接受审核的用户。

在已转换为受信任模式的系统上，将给每个登录会话分配一个名为 “审核标记” 的唯一标识符。 “审核标记” 是一个表示诸如用户名和登录时间等信息的字符串。它可以唯一地识别每次登录会话和该会话的负责人。另请参阅 *setauduser*(3) 和 *getauduser*(3)。 **userdbset** 命令用于指定那些要接受审核的用户。请参阅 *userbdset*(1M) 和 *userdb*(4)。相关的属性称为 **AUDIT_FLAG** ，并在 *security*(4)中进行了说明。

audevent 命令用于指定要审核的系统活动（可审核的事件）。为了便于配置，可审核的事件分为几个事件类别和配置文件。一旦选择某个事件类别或配置文件，就选择了与该事件类别或配置文件关联的所有系统调用和自审核事件。在安装审核系统时，在文件 **/etc/audit/audit.conf** 中提供了一组缺省的事件类别信息。为了满足站点特定的要求，管理员还可以在 **/etc/audit/audit_site.conf** 中定义事件类别和配置文件。有关详细信息，请参阅 *audit.conf*(4) 和 *audevent*(1M) 。

请注意，即使未选择用户进行审核，在用户启动会话和结束会话时也应生成一些记录。这些记录被视为事件选择（而非用户选择）的系统范围的信息。其他进行自审核的程序也可能会执行自己的决定而忽略用户的选择，尽管建议不这样执行。可在稍后获得有关自审核的详细信息。

开始和暂停审核系统

管理员可以使用 **audsys** 命令开始或暂停审核系统，或获取审核系统状况的简短摘要。开始审核系统之前， **audsys** 也验证指定的参数，并确保审核系统处于安全和一致的状态。有关详细信息，请参阅 *audsys*(1M) 。

监视审核系统

为了确保审核系统工作正常并检测任何不正常行为，可在后台运行具有权限的 守护程序 **audomon** ，以监视各种审核系统参数。当这些参数引用了不正常（危险）值，或审核系统的组件被意外删除时， **audomon** 将输出警告消息并在可能条件下尝试解决问题。有关详细信息，请参阅 *audomon*(1M) 。如果在 **/etc/rc.config.d/auditing** 文件中将参数 *AUDITING* 设置为 1，则在系统引导时， **audomon** 可以由 **/sbin/init.d/auditing** 衍生（作为 **init** 启动进程的一部分）。它还可以由具有权限的用户随时启动。

查看审核数据

audisp 命令用于查看日志文件中记录的审核数据。 **audisp** 命令将日志文件按时间顺序合并为一个审核记录。管理员可以选择由 **audisp** 命令提供的查看标准，将搜索范围限制在管理员有兴趣调查的特定类型的事件。

审核记录

在启用审核系统的任何时候，必须至少提供一个审核记录。可使用 **audsys** 来指定记录的名称和各种属性。当前记录超过指定的大小时，或者审核文件系统有被填满的危险时，系统会自动切换到另一个记录（该记录具有相同的基名，但是时间戳扩展名不同）并开始记录到该记录。在每次成功转换后，可以使用 **audomon** 指定一个脚本执行最后一个审核记录上的各种操作。如果记录切换不成功，则会发送警告消息，请求管理员采取相应的操作。

自审核程序

为了减少日志的数据量并提供对某些典型系统操作的高级记录，为一组具有权限的程序赋予了执行自审核的功能。这意味着程序自身可以挂起当前指定的审核并生成对其执行操作的高级说明。这些自审核程序在下列联机帮助页中进行了详细说明： *at*(1)、 *chfn*(1)、 *chsh*(1)、 *crontab*(1)、 *login*(1)、 *newgrp*(1)、 *passwd*(1)、 *audevent*(1M)、 *audisp*(1M)、 *audsys*(1M)、 *audusr*(1M)、 *cron*(1M)、 *groupadd*(1M)、 *groupdel*(1M)、 *groupmod*(1M)、 *init*(1M)、 *lpsched*(1M)、 *sam*(1M)、 *useradd*(1M)、 *userdel*(1M) 和 *usermod*(1M)。

> 注释：仅允许具有权限的程序进行自审核。它们挂起所执行的审核，仅仅影响这些程序而不影响系统中的任何其他进程。

大多数这些命令在单一事件类别下生成审核数据。例如， **SAM** 在 *admin* 事件下生成审核数据。其他命令可以在多个事件类别下生成数据。例如， **init** 命令可在事件 *login* 和 *admin* 下生成数据。有关预定义的事件类别的列表，请参阅 *audevent(1M)*。

警告

HP-UX 11i v3 是支持受信任系统功能的最后一个版本。

HP-UX 审核系统继续运行，不需要转换为受信任模式。

作者

上述审核系统由 HP 开发。

另请参阅

audevent(1M)、 audisp(1M)、 audsys(1M)、 audusr(1M)、 userdbset(1M)、 audctl(2)、 audswitch(2)、 audwrite(2)、 getaudid(2)、 getevent(2)、 setaudid(2)、 setevent(2)、 getauduser(3)、 setauduser(3)、 audit(4)、 security(4)、 userdb(4)、 audit_memory_usage(5)、 audit_track_paths(5)、 diskaudit_flush_interval(5)。

名称

audit_memory_usage - 审核子系统可以使用的物理内存的百分比

值

保证安全

5

缺省值

5

允许值

1 - 10

建议值

5

说明

可调参数 **audit_memory_usage** 定义审核记录可以使用的物理内存的百分比。如果审核记录内存使用量超过 **audit_memory_usage** 限制，审核子系统将阻塞系统调用，直到内存使用量降到该限制范围之内。

更改此可调参数的人员

具有相应权限的管理员可以更改 **audit_memory_usage** 的值。

更改限制

由于该可调参数是一个动态可调参数，因此可以随时更改它的值；但它的值不能低于审核子系统当前用于记录的内存量。

应该何时增加此可调参数的值

当要审核的系统非常重要并生成许多记录时。

增加此值的负面影响

如果系统生成大量记录，那么增加该值可以产生大量未刷新的内存，并可能降低系统的速度。

应该何时降低此可调参数的值

当审核子系统不生成大量记录时。

降低此值的负面影响

如果生成大量审核记录，则审核子系统可能会阻塞系统调用。

应该同时更改的其他可调参数值

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是

缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

audit_memory_usage 由 HP 开发。

另请参阅

audit(5)。

名称

audit_track_paths - 启用（或禁用）对当前目录和根目录的跟踪以审核子系统

值

保证安全

0（禁用）

缺省值

0（禁用）

允许值

0（禁用）或 **1**（启用）

建议值

如果已启用 **Audit** 或已安装 **HP-UX HIDS** ，则为 **1** （启用），

否则为 **0**（禁用）。

说明

audit_track_paths 为动态可调参数，并且替换了 **HP-UX HIDS** 特定静态可调参数 **enable_idds** 。

将可调参数 **audit_track_paths** 设置为 **1** ，可使 **Audit** 和 **HP-UX HIDS** 解析并报告用于记账目的的绝对路径名称。这也会导致内核的其他跟踪，从而使性能略微降低（同时使用更多内核内存），即使没有使用审核子系统时也是如此。尽管不是必需的，但是强烈建议在将可调参数 **audit_track_paths** 设置为 **1** 时重新引导系统，以便能够记录绝对路径名称。否则， **Audit** 或 **HP-UX HIDS** 不能稳定地解析和报告绝对路径名称。

audit_track_paths 设置为 **0** 时， **Audit** 将不解析绝对路径名称，同时， **HP-UX HIDS** 将无法打开设备并收集数据。这是因为 HIDS 总是期望一个完整路径名称。

如果安装系统时尚未安装 **HP-UX HIDS** ，则该可调参数设置为 **Default** 状态，并且其值设置为 **0** 。如果首先安装了 **HP-UX HIDS** ，则该可调参数设置为 **1** 。

更改此可调参数的人员

具有适当权限的管理员可以根据下面说明的限制，更改 **audit_track_paths** 的值。

更改限制

可调参数 **audit_track_paths** 是动态可调参数，所以在满足以下条件时，对此参数的所有更改将立即生效：

1) 如果新的可调参数值为 0（非 **Default** ），则 **HPUX HIDS** 将无法打开 IDDS 设备；因此，也将无法运行需要系统调用审核记录的任意入侵检测模板。强制此限制用于防止 HIDS 报告不完整的路径名或相对路径名。
2) 如果 **/dev/idds** 已打开，则不允许管理员更改可调参数的值。
3) 如果将可调参数设置为 **Default** ，则 **HPUX HIDS** 打开 IDDS 设备时， **IDDS** 将自我调整其值为 **1** 。
4) 如果将可调参数值设置为 **Default** ，则审核设置为 **ON** 时， **Audit** 将自我调整其值为 **1** 。

5) 如果 **Audit** 已经设置为 **ON** ，则不允许管理员更改可调参数值。

6) 如果管理员将可调参数值从 **0** 更改为 **1** ，则建议用户重新引导系统，以避免 **HP-UX HIDS** 或 **Audit** 报告部分路径名。

应该何时启用此可调参数选项

如果要启动 **HP-UX HIDS** 或 **Audit** ，则可调参数 **audit_track_paths** 应当设置为 **ON** 。

启用此可调参数的负面影响

跟踪每个进程的当前工作目录（以及根目录）的名称，从而使内存使用和系统性能发生更改。

应该何时禁用此可调参数选项

HIDS 和 **Audit** 都设置为 **OFF** 时。

禁用此可调参数的负面影响

可调参数设置为 **OFF** 时， **HP-UX HIDS** 无法使用任何需要系统调用审核记录的检测模板（如“文件或目录修改模板”）。有关模板的详细信息，请参阅 HP-UX HIDS 文档。同时在此情况下， **Audit** 将在审核日志中报告相对路径名称。

应该同时更改的其他可调参数

此可调参数与其他可调参数无关。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

audit_track_paths 由 HP 开发。

另请参阅

kctune(1M)、 audit(5)、 ids.cf(5)。

名称

chanq_hash_locks - 保护通道队列散列表的 spinlock 散列池的大小

值

无故障

256

缺省值

256

允许值

16 和 **16777216** 之间为 2 的幂的任意值。

建议值

只能是缺省值。

HP 不建议更改此可调参数的缺省值。更改此可调参数前请咨询 HP 现场服务工程师。

说明

不应更改此可调参数。缺省值是优化系统性能的最好选择。

此可调参数控制 spinlock 池（用于同步的内核数据结构）的大小，也保护通道队列。过小的值会增加通道队列散列表中冲突并降低性能的可能性。过大的值则不必要。

应该由谁来更改此可调参数?

HP 不建议更改此可调参数的缺省值。更改此可调参数前请咨询用户的 HP 现场服务工程师。

对于更改的限制

此可调参数应该是 2 的幂。如果指定了其他值，那么使用 2 的幂的最大值（小于给定值）。

增加或降低此值的副作用是什么?

系统性能会下降。

同时还应更改哪些其他可调参数的值?

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。未来的 HP-UX 版本可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

chanq_hash_locks 由 HP 开发。

名称

compartments - HP-UX 隔离专区说明

说明

UNIX® 操作系统通常使用一个隔离专区模型。由于传统的单个隔离专区系统中的访问相对比较自由，因此可能会因恶意软件或受到安全威胁的程序而产生一些问题。如果发现并使用了一种利用守护程序进程的方法，则入侵者将获得相当高的系统访问权限。如果守护程序进程在通过有效 uid **0** 运行的同时被利用，则该权限可能转换为完全的系统访问权限。使用隔离专区，可以仅限于对进程所需的内容进行访问，从而降低恶意程序或被利用程序可能造成的破坏程度。

隔离专区将某个进程隔离，使其只能访问同一隔离专区中的对象，除非隔离专区规则向该进程授予其他隔离专区的访问权限。其他访问控制方法（如文件权限和 ACL）仍将适用。

可以使用相应的权限覆盖隔离专区限制。有关权限列表，请参阅 *privileges*(5) 。

隔离专区控制进程对多种不同类型的系统对象的访问权限。其中的某些对象类型是持久性的，并通常按名称（如文件）引用。这些对象没有与其直接关联的隔离专区。而是用于控制这些对象访问权限的规则与对象名称关联。其他对象类型是瞬态的，其持续时间仅与创建它们的进程相同，或与系统引导时间相同。瞬态对象与创建它们的进程的隔离专区一同被标记。用于控制这些对象访问权限的规则是一个直接的隔离专区对隔离专区关系。

隔离专区控制三种类型的系统对象：文件系统对象（持久对象）、进程间通信 (IPC) 对象（瞬态对象）、网络对象（瞬态对象）：

- 文件系统对象。包括文件和目录。缺省情况下，任何隔离专区均可以访问所有文件系统对象。但是，特定隔离专区配置可以定义规则来限制对各种文件系统对象的访问权限。
- 进程间通信 **(IPC)** 对象。支持或限制在单个系统上的进程之间进行通信。 IPC 对象类型包括 System V 共享内存、 System V 信号量、 System V 消息队列、 POSIX 信号量、 POSIX 消息队列、 PTY、 FIFO、 UNIX 域套接字以及进程（信号机制）。 POSIX 共享内存作为文件系统对象实现，因此隔离专区访问权限通过文件系统规则进行控制。除非另外进行了显式配置，否则缺省情况下，给定隔离专区中的进程无法访问其他隔离专区中的 IPC 对象。
- 网络通信对象。包括网络端点（套接字和流）以及局域网接口。这些对象用于通过 TCP/IP 协议与本地系统和远程系统上的进程通信。进程的网络端点与局域网接口（流量通过该接口传递到远程系统）之间的访问受到控制。与 IPC 对象一样，除非进行了显式配置，否则给定隔离专区中的进程无法访问其他隔离专区中的网络对象。

  每个网络局域网接口（逻辑/物理/虚拟）可以属于其自己的隔离专区。例如，可以对规则进行设置，使逻辑接口 **lan0:1** 和 **lan0:2** 属于不同的隔离专区。

配置规则

在系统启动时，从 **/etc/cmpt** 目录中的文件读取隔离专区配置。配置放置在 **/etc/cmpt** 下以 **.rules** 后缀结尾的文件中。这些文件在应用前通过 **cpp** 进行预处理。可以使用 **cpp** 的机制（如 C/C++ 样式注释、 **#ifdef** 和 **#include** ）来组织这些文件。有关配置文件的语法，请参阅 *compartments*(4) 。

隔离专区使用四种类型的规则：文件系统规则、IPC 规则、网络规则、其他规则。

文件系统规则

文件系统规则控制文件系统的文件和目录的访问权限。可以将目录访问限制为下列操作：

- **nsearch**： 用于搜索目录。
- **read**： 用于目录列举和搜索
- **create**： 用于在目录下创建新元素
- **unlink**： 用于删除目录下的元素
- 以上四个操作的任意组合

可以将文件访问限制为下列操作：

- **read**： 用于读取或执行文件
- **write**： 用于写入文件
- 以上两个操作的任意组合

除 **nsearch** 访问之外，将继承所有文件系统规则。例如，如果 **/a** 拥有 **nsearch** 和 **create** 的权限，则除非向其分配其他权限集，否则 **/a/b** 将单独拥有 **create** 权限。

IPC 规则

IPC 规则控制该隔离专区中的进程如何访问其他隔离专区的 IPC 机制，以及其他隔离专区中的进程如何访问该隔离专区的 IPC 机制。缺省情况下，进程只能访问它自己的隔离专区中的 IPC 对象。

网络规则

网络规则控制进程与网络接口之间以及两个使用环回通信的进程之间的访问。这些规则控制规则中指定的主题隔离专区与目标隔离专区之间的网络流量的方向（传入、传出或双向）。每个规则都指定流量方向、协议（TCP、UDP 或原始协议）以及目标隔离专区（针对网络接口或用于本地进程通信的本地隔离专区）。规则也可以针对本地端口号和对等端口号进行过滤（仅限 TCP 和 UDP）。

最初创建时隔离专区即与网络端点相关联。当进程发出创建端点的系统调用（ **socket()** 或 **open()** ）时，该时刻的进程的隔离专区将应用于网络对象。（请参阅 *socket*(2) 或 *open*(2) ）。该隔离专区用于所有涉及该对象的网络通信访问检查。对于 TCP，在建立连接时应用规则。对于所有其他网络通信，将针对规则检查每个入站和出站数据包传递。

其他规则

其他规则出现在隔离专区定义中。这些规则包括以下项：

禁用的权限

禁用的权限定义由于 **exec()** 调用的负面影响而无法获取的特定权限（即使执行的二进制文件指定权限变为可用）。请参阅 *exec*(2) 。请参阅 **setfilexsec** 命令的 **-p** 和 **-r** 标记的说明。有关进程如何在 **exec()** 调用的负面影响下获取权限的信息，请参阅 *setfilexsec*(1M) 。

网络接口规则

接口规则定义该隔离专区中包含哪些网络接口（物理/虚拟/逻辑）。尽管可以将多个接口分配给同一隔离专区，但每个网络接口只能属于一个隔离专区。另外请注意，某些特殊的逻辑接口（如环回接口 **lo0** 和通道接口）不是有效的配置参数。这些接口将以无提示方式被忽略。

与隔离专区相关的权限

以下权限集（请参阅 *privileges*(5)）影响隔离专区的操作：

CHANGECMPT 使进程可以更改其隔离专区。

CMPTREAD 允许进程打开文件或目录进行读取、执行（针对文件）或搜索（针对目录），从而绕过在其他情况下本不允许执行该操作的隔离专区规则。

CMPTWRITE 允许进程写入文件，或在目录中创建或删除文件，从而绕过在其他情况下本不允许执行该操作的隔离专区规则。

COMMALLOWED 允许进程覆盖隔离专区 IPC 和网络规则。

RULESCONFIG 允许进程修改系统中的隔离专区规则。

注释：缺省情况下，这些权限不会自动授予具有有效 uid **0** 的进程。

缺省隔离专区

将隔离专区安装到系统后，只存在一个缺省隔离专区，即 **init** 隔离专区。当系统引导时， **init** 进程将属于此隔离专区。该隔离专区已经定义为可以访问为系统显式定义的所有其他隔离专区。不必在规则文件中定义 **init** 隔离专区。如果重新定义了 **init** 隔离专区（方法是在规则文件中显式引用它），则将丢失所有特殊特性，并只能在重新引导系统后恢复这些特性。

隔离专区操作命令

有多个命令可用于在系统上检查和修改隔离专区配置：

cmpt_tune 查询、启用和禁用隔离专区功能。有关详细信息，请参阅 *cmpt_tune*(1M)。

getrules 显示隔离专区规则。有关详细信息，请参阅 *getrules*(1M)。

setrules 分析并实现规则。有关详细信息，请参阅 *setrules*(1M)。

注释：当前没有可用于修改隔离专区配置文件的命令。必须直接编辑配置文件。完成编辑后，可以使用以上命令实现这些配置文件。

文件

/etc/cmpt/rules/ 该目录下所有名称以 **.rules** 结尾的文件均用于创建隔离专区配置。所有用于在系统上配置隔离专区规则的文件（由 **#include** 指令引用的文件除外）必须位于此目录中。

/etc/cmpt-rules.bin 包含计算机可读隔离专区规则的二进制文件。请 不要直接编辑此文件。

/etc/cmpt-db 将隔离专区名称映射到系统内部使用的 ID 号的文件。请 不要直接编辑此文件。

另请参阅

cmpt_tune(1M)、 getrules(1M)、 setrules(1M)、 exec(2)、 open(2)、 socket(2)、 compartments(4)、 privileges(5)。

名称

complex - 复数函数和宏

概要

#include <complex.h>

说明

本文件包含复数库（在第 (3M) 节中描述）中所有函数的声明。

它定义了类型

extended Integrity 服务器 80 位双精度扩展型。

quad 符合 IEEE 754 标准的、 128 位浮点型。在 HP-UX 上， **quad** 与 **long double** 同义。

它定义了宏

complex 扩展为 **_Complex** ，指定复数类型的关键字。

imaginary 扩展为 **_Imaginary** ，指定虚数类型的关键字。

这些宏以及实际类型的名称组合起来，提供了对复数和虚数类型的指定

float complex
double complex
long double complex
extended complex
quad complex
float imaginary
double imaginary
long double imaginary
extended imaginary
quad imaginary

本文件也定义了宏

_Imaginary_I 扩展为带有虚部值（平方为 -1 的数）的 **const float imaginary** 类型的常量表达式。

_Complex_I 扩展为带有虚部值的 **const float complex** 类型的常量表达式。

I 扩展为 **_Imaginary_I** 。它可以用于有效地构造 **x + y*I** 样式的复数表达式，其中 **x** 和 **y** 是实数。

此头文件仅适用于 Integrity 服务器。要使用它，请使用缺省的 **-Ae** 选项进行编译。要使用类型 **extended complex**、 **quad complex** 、 **extended imaginary** 或 **quad imaginary** ，则使用 **-fpwidetypes** 选项进行编译。

文件

/usr/include/complex.h

另请参阅

intro(3)、 fenv(5)、 math(5)。

符合的标准

<complex.h>： ISO/IEC C99（包括附件 G： “IEC 60559-compatible complex arithmetic”）

名称

core_addshmem_read - 确定在进程核心转储中包含可读共享内存

值

无故障

0 （关闭）

缺省值

0 （关闭）

允许值

1 （打开）或 **0** （关闭）

说明

添加 **core_addshmem_read** 可调参数响应由于共享内存段不是写为核心文件的一部分而造成的调试用户级进程崩溃中客户受限的问题。

核心转储时，如果此可调参数设置为 **1** （打开），则标记为只读共享的用户内存部分被写入（连同普通数据部分），如果它设置为 **0** （关闭）则不用考虑。

应该由谁来更改此可调参数？

任何人。

对于更改的限制

对此可调参数的更改立即生效。

何时应增加此可调参数的值？

当由于开发人员或系统维护人员需要在系统上调试而希望在用户核心文件中包含只读共享内存段时。

增加此值的副作用是什么？

在大多数情况（大多数应用程序至少使用一些共享内存）下用户进程核心文件会增加。当磁盘空间紧张的情况下这会成为系统的一个问题。

何时应降低此可调参数的值？

当不需要调试共享内存损坏或数据值有疑问的核心文件时。

减小此值的副作用是什么？

核心文件会变小。

同时还应更改哪些其他可调参数的值？

当然要考虑 **core_addshmem_write** ，该值对读取/写入共享内存段进行相同的操作。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。未来的 HP-UX 版本可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统

上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

core_addshmem_read 由 HP 开发。

名称

core_addshmem_write - 确定在进程核心转储中包含读取/写入共享内存

值

无故障

0 （关闭）

缺省值

0 （关闭）

允许值

1 （打开）或 **0** （关闭）

说明

添加 **core_addshmem_write** 可调参数响应由于共享内存段不是写为核心文件的一部分而造成的调试用户级进程崩溃中客户受限的问题。

核心转储时，如果此可调参数设置为 **1** （打开），则标记为读写共享的用户内存部分被写入（连同普通数据部分），如果它设置为 **0** （关闭）则不用考虑。

应该由谁来更改此可调参数？

任何人。

对于更改的限制

对此可调参数的更改立即生效。

何时应增加此可调参数的值？

当由于开发人员或系统维护人员需要在系统上调试而希望在用户核心文件中包括读写共享内存段时。

增加此值的副作用是什么？

在大多数情况（大多数应用程序至少使用一些共享内存）下用户进程核心文件会增加。当磁盘空间紧张的情况下这会成为系统的一个问题。

何时应降低此可调参数的值？

当不需要调试共享内存损坏或数据值有疑问的核心文件时。

降低此值的副作用是什么？

核心文件会变小。

同时还应更改哪些其他可调参数的值？

当然要考虑 **core_addshmem_read** ，该值对只读共享内存段进行相同的操作。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。未来的 HP-UX 版本可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统

上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

core_addshmem_write 由 HP 开发。

名称

create_fastlinks - 配置系统以使用快速符号链接

值

最小值

0

最大值

1

缺省值

0

指定整数值。

说明

当 **create_fastlinks** 为非零时，它使系统通过对路径名查找中每个符号链接的磁盘块访问次数减一的方式，创建 HFS 符号链接。这涉及到 HFS 磁盘格式的细微更改，导致任何为了快速符号链接格式化的磁盘在 HP-UX 9.0 之前的 700 系列系统和 HP-UX 10.0 之前的 800 系列系统上不能使用（该可配置参数存在于 9.0 发行版 700 系列系统上，但不在 HP-UX 9.0 800 系列系统上）。

要提供向后兼容性， **create_fastlinks** 的缺省设置为零，这样不会创建更新、更快的格式。然而，所有 HP-UX 10.0 内核（以及所有 HP-UX 9.0 700 系列内核）对两种磁盘格式都可识别，无论“create_fastlinks”设置为零或非零。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。未来的 HP-UX 版本可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

create_fastlinks 由 HP 开发。

名称

curses： curses.h - 定义屏幕处理和优化函数

概要

#include <curses.h>

说明

对象

<curses.h> 头文件提供了 *COLOR_PAIRS* 、 *COLORS* 、 *COLS* 、 *curscr* 、 *LINES* 和 *stdscr* 的声明。

常量

定义了下列常量:

EOF 文件结束函数返回值

ERR 失败函数返回值

FALSE 布尔型 *false* 值

OK 成功函数返回值

TRUE 布尔型 *true* 值

WEOF 文件结束的宽字符函数返回值，在 **<wchar.h>** 中定义。

数据类型

下列数据类型通过 **typedef** 定义:

attr_t 属性的 OR 集合

bool 布尔型数据类型

chtype 字符、属性 和色彩对

SCREEN 不透明终端形式

wchar_t 如 **<stddef.h>** 中所述

wint_t 如 **<wchar.h>** 中所述

cchar_t 引用宽字符字符串

WINDOW 不透明窗口形式

在 *curses_intro*(3X) 的“数据类型”中更详细地说明了这些数据类型。

<curses.h> 包括的内容可以使头文件 **<stdio.h>** 、 **<term.h>** 、 **<termios.h>** 和 **<wchar.h>** 中所有的符号可见。

属性位

下列符号常量用于处理 **attr_t** 类型对象:

WA_ALTCHARSET	其他字符集
WA_BLINK	闪烁
WA_BOLD	加亮或加粗
WA_DIM	半亮
WA_HORIZONTAL	横向高亮
WA_INVIS	不可见
WA_LEFT	左边高亮
WA_LOW	降低高亮
WA_PROTECT	保护
WA_REVERSE	反向视频
WA_RIGHT	右边高亮
WA_STANDOUT	终端的最佳高亮模式
WA_TOP	顶部高亮
WA_UNDERLINE	下划线
WA_VERTICAL	垂直高亮

这些属性标记应该不同。

下列符号常量用于处理类型是 **chtype** 的对象的属性位:

A_ALTCHARSET	其他字符集
A_BLINK	闪烁
A_BOLD	加亮或加粗
A_DIM	半亮
A_INVIS	不可见
A_PROTECT	保护
A_REVERSE	反向视频
A_STANDOUT	终端的最佳高亮模式
A_UNDERLINE	下划线

这些属性标记不必一定是不同的, 除非定义了 _XOPEN_CURSES 并且应用程序将 _XOPEN_SOURCE_EXTENDED 设置为 1。

下列符号常量可用做位掩码来提取 **chtype** 的组成部分:

A_ATTRIBUTES	提取属性的位掩码
A_CHARTEXT	提取字符的位掩码
A_COLOR	提取色彩对信息的位掩码

下列符号常量可用做位掩码来提取 **chtype** 的组成部分:

A_ATTRIBUTES	提取属性的位掩码
A_CHARTEXT	提取字符的位掩码
A_COLOR	提取色彩对信息的位掩码

线条绘制常量

<curses.h> 头文件定义下表中最左侧两列显示的符号常量，用于绘制线条。以 ACS_ 开头的符号常量是 **char** 常量。以 WACS_ 开头的符号常量是 **cchar_t** 常量，用于宽字符接口，该宽字符接口使用指向 **cchar_t** 的指针。

POSIX 语言环境中，当终端数据库未使用如 *terminfo*(4) 中的“Line Graphics”所述的 **acsc** 功能指定值时，使用“POSIX Locale Default”列显示的字符。

字符常量	**cchar_t** 常量	**POSIX** 语言环境 缺省值	符号说明
ACS_ULCORNER	WACS_ULCORNER	+	左上角
ACS_LLCORNER	WACS_LLCORNER	+	左下角
ACS_URCORNER	WACS_URCORNER	+	右上角
ACS_LRCORNER	WACS_LRCORNER	+	右下角
ACS_RTEE	WACS_RTEE	+	右 T 型
ACS_LTEE	WACS_LTEE	+	左 T 型
ACS_BTEE	WACS_BTEE	+	下 T 型
ACS_TTEE	WACS_TTEE	+	上 T 型
ACS_HLINE	WACS_HLINE	-	水平线
ACS_VLINE	WACS_VLINE	|	垂直线
ACS_PLUS	WACS_PLUS	+	加
ACS_S1	WACS_S1	-	扫描线 1
ACS_S9	WACS_S9	_	扫描线 9
ACS_DIAMOND	WACS_DIAMOND	+	菱形
ACS_CKBOARD	WACS_CKBOARD	:	检测板（点画）
ACS_DEGREE	WACS_DEGREE	'	程度符号
ACS_PLMINUS	WACS_PLMINUS	#	加/减
ACS_BULLET	WACS_BULLET	o	项目符号
ACS_LARROW	WACS_LARROW	<	左箭头
ACS_RARROW	WACS_RARROW	>	右箭头
ACS_DARROW	WACS_DARROW	v	向下箭头
ACS_UARROW	WACS_UARROW	^	向上箭头
ACS_BOARD	WACS_BOARD	#	方形板
ACS_LANTERN	WACS_LANTERN	#	罩符号
ACS_BLOCK	WACS_BLOCK	#	实心方块

色彩相关宏

定义下列色彩相关宏：

COLOR_BLACK
COLOR_BLUE
COLOR_GREEN
COLOR_CYAN
COLOR_RED
COLOR_MAGENTA
COLOR_YELLOW
COLOR_WHITE

坐标相关宏

定义下列坐标相关宏：

void getbegyx(WINDOW ***win*, int *y*, int *x*);**
void getmaxyx(WINDOW ***win*, int *y*, int *x*);**
void getparyx(WINDOW ***win*, int *y*, int *x*);**
void getyx(WINDOW ***win*, int *y*, int *x*);**

按键代码

定义下列表示函数按键值的符号常量：

按键代码	说明
KEY_CODE_YES	用于指示 **wchar_t** 变量包含按键代码
KEY_BREAK	Break 键
KEY_DOWN	向下键
KEY_UP	向上键
KEY_LEFT	向左键
KEY_RIGHT	向右键
KEY_HOME	Home 键
KEY_BACKSPACE	退格键
KEY_F0	功能键；64 键的空间保留
KEY_F(*n*)	$0 \le n \le 63$
KEY_DL	删除行
KEY_IL	插入行
KEY_DC	删除字符
KEY_IC	插入字符或进入插入模式
KEY_EIC	退出插入字符模式
KEY_CLEAR	清除屏幕
KEY_EOS	清除到屏幕结尾

KEY_EOL	将光标移到行的结尾。
KEY_SF	向前滚动 1 行
KEY_SR	向后滚动 1 行（逆向）
KEY_NPAGE	下一页
KEY_PPAGE	上一页
KEY_STAB	设置制表位
KEY_CTAB	清除制表位
KEY_CATAB	清除所有制表位
KEY_ENTER	回车或发送
KEY_SRESET	软（部分）重置
KEY_RESET	重置或硬重置
KEY_PRINT	打印或复制
KEY_LL	向下翻页或底部
KEY_A1	键盘左上角
KEY_A3	键盘右上角
KEY_B2	键盘中心
KEY_C1	键盘左下角
KEY_C3	键盘右下角

虚拟键盘为 3×3 键盘，排列如下：

A1	Up	A3
Left	B2	Right
C1	Down	C3

每个图标符号，如 A1，对应着前表中的一个按键代码的符号常量，如 KEY_A1。下列表示函数按键值的符号常量也被定义：

按键代码	说明
KEY_BTAB	回退制表位键
KEY_BEG	开始键
KEY_CANCEL	取消键
KEY_CLOSE	关闭键
KEY_COMMAND	Cmd（命令）键
KEY_COPY	复制键
KEY_CREATE	创建键
KEY_END	结束键
KEY_EXIT	退出键
KEY_FIND	查找键

KEY_HELP	帮助键
KEY_MARK	标记键
KEY_MESSAGE	消息键
KEY_MOVE	移动键
KEY_NEXT	下一对象键
KEY_OPEN	打开键
KEY_OPTIONS	选项键
KEY_PREVIOUS	前一对象键
KEY_REDO	重复操作键
KEY_REFERENCE	引用键
KEY_REFRESH	刷新键
KEY_REPLACE	替换键
KEY_RESTART	重新启动键
KEY_RESUME	恢复键
KEY_SAVE	保存键
KEY_SBEG	换档开始键
KEY_SCANCEL	换档取消键
KEY_SCOMMAND	换档命令键
KEY_SCOPY	换档复制键
KEY_SCREATE	换档创建键
KEY_SDC	换档删除字符键
KEY_SDL	换档删除行键
KEY_SELECT	选择键
KEY_SEND	换档到结束键
KEY_SEOL	换档清除行键
KEY_SEXIT	换档退出键
KEY_SFIND	换档查找键
KEY_SHELP	换档帮助键
KEY_SHOME	换档 home 键
KEY_SIC	换档输入键
KEY_SLEFT	换档向左键
KEY_SMESSAGE	换档消息键
KEY_SMOVE	换档移动键
KEY_SNEXT	换档下一键
KEY_SOPTIONS	换档选项键
KEY_SPREVIOUS	换档先前键
KEY_SPRINT	换档打印键
KEY_SREDO	换档重复操作键

KEY_SREPLACE	换档替换键
KEY_SRIGHT	换档右箭头
KEY_SRSUME	换档恢复键
KEY_SSAVE	换档保存键
KEY_SSUSPEND	换档挂起键
KEY_SUNDO	换档取消操作键
KEY_SUSPEND	挂起键
KEY_UNDO	取消操作键

函数原型

下列内容可以声明为函数，也可以定义为宏:

```
int   addch(const chtype ch);
int   addchstr(const chtype *chstr);
int   addchnstr(const chtype *chstr, int n);
int   addnstr(const char *str, int n);
int   addstr(const char *str);
int   addnwstr(const wchar_t *wstr, int n);
int   addwstr(const wchar_t *wstr);
int   add_wch(const cchar_t *wch);
int   add_wchnstr(const cchar_t *wchstr, int n);
int   add_wchstr(const cchar_t *wchstr);
int   attroff(int attrs);
int   attron(int attrs);
int   attrset(int attrs);
int   attr_get(attr_t *attrs, short *color_pair, void *opts);
int   attr_off(attr_t attrs, void *opts);
int   attr_on(attr_t attrs, void *opts);
int   attr_set(attr_t attrs, short color_pair, void *opts);
int   baudrate(void);
int   beep(void);
int   bkgd(chtype ch);
void  bkgdset(chtype ch);
void  bkgrndset(const cchar_t *wch);
int   bkgrnd(const cchar_t *wch);
int   border(chtype ls, chtype rs, chtype ts, chtype bs, chtype tl,
          chtype tr, chtype bl, chtype br);
int   border_set(const cchar_t *ls, const cchar_t *rs,
            const cchar_t *ts, const cchar_t *bs,
            const cchar_t *tl, const cchar_t *tr,
```

```
                const cchar_t *bl, const cchar_t *br);
int     box(WINDOW *win, chtype verch, chtype horch);
int     box_set(WINDOW *win, const cchar_t *verch,
             const cchar_t *horch);
bool    can_change_color(void);
int     cbreak(void);
int     chgat(int n, attr_t attr, short color, const void *opts);
int     clear(void);
int     clearok(WINDOW *win, bool bf);
int     clrtobot(void);
int     clrtoeol(void);
int     color_content(short color, short *red, short *green,
                short *blue);
int     COLOR_PAIR(int n);
int     copywin(const WINDOW *srcwin, WINDOW *dstwin, int sminrow,
             int smincol, int dminrow, int dmincol, int dmaxrow,
             int dmaxcol, int overlay);
int     curs_set(int visibility);
int     def_prog_mode(void);
int     def_shell_mode(void);
int     delay_output(int ms);
int     delch(void);
void    delscreen(SCREEN *sp);
int     delwin(WINDOW *win);
int     deleteln(void);
WINDOW *derwin(WINDOW *orig, int nlines, int ncols, int begin_y,
             int begin_x);
int     doupdate(void);
WINDOW *dupwin(WINDOW *win);
int     echo(void);
int     echochar(const chtype ch);
int     echo_wchar(const cchar_t *wch);
int     endwin(void);
int     erase(void);
char    erasechar(void);
int     erasewchar(wchar_t *ch);
void    filter(void);
int     flash(void);
int     flushinp(void);
```

```
chtype getbkgd(WINDOW *win);
int    getbkgrnd(cchar_t *wch);
int    getcchar(const cchar_t *wcval, wchar_t *wch, attr_t *attrs,
               short *color_pair, void *opts);
int    getch(void);
int    getnstr(char *str, int n);
int    getn_wstr(wint_t *wstr, int n);
int    getstr(char *str);
int    get_wch(wint_t *ch);
WINDOW *getwin(FILE *filep);
int    get_wstr(wint_t *wstr);
int    halfdelay(int tenths);
bool   has_colors(void);
bool   has_ic(void);
bool   has_il(void);
int    hline(chtype ch, int n);
int    hline_set(const cchar_t *wch, int n);
void   idcok(WINDOW *win, bool bf);
int    idlok(WINDOW *win, bool bf);
void   immedok(WINDOW *win, bool bf);
chtype inch(void);
int    inchnstr(chtype *chstr, int n);
int    inchstr(chtype *chstr);
WINDOW *initscr(void);
int    init_color(short color, short red, short green, short blue);
int    init_pair(short pair, short f, short b);
int    innstr(char *str, int n);
int    innwstr(wchar_t *wstr, int n);
int    insch(chtype ch);
int    insdelln(int n);
int    insertln(void);
int    insnstr(const char *str, int n);
int    insstr(const char *str);
int    instr(char *str);
int    ins_nwstr(const wchar_t *wstr, int n);
int    ins_wch(const cchar_t *wch);
int    ins_wstr(const wchar_t *wstr);
int    intrflush(WINDOW *win, bool bf);
int    in_wch(cchar_t *wcval);
```

int in_wchstr(cchar_t **wchstr*);

int in_wchnstr(cchar_t **wchstr*, int *n*);

int inwstr(wchar_t **wstr*);

bool isendwin(void);

bool is_linetouched(WINDOW **win*, int *line*);

bool is_wintouched(WINDOW **win*);

char *keyname(int *c*);

char *key_name(wchar_t *c*);

int keypad(WINDOW **win*, bool *bf*);

char killchar(void);

int killwchar(wchar_t **ch*);

int leaveok(WINDOW **win*, bool *bf*);

char *longname(void);

int meta(WINDOW **win*, bool *bf*);

int move(int *y*, int *x*);

int mvaddch(int *y*, int *x*, const chtype *ch*);

int mvaddchnstr(int *y*, int *x*, const chtype **chstr*, int *n*);

int mvaddchstr(int *y*, int *x*, const chtype **chstr*);

int mvaddnstr(int *y*, int *x*, const char **str*, int *n*);

int mvaddnwstr(int *y*, int *x*, const wchar_t **wstr*, int *n*);

int mvaddstr(int *y*, int *x*, const char **str*);

int mvaddwstr(int *y*, int *x*, const wchar_t **wstr*);

int mvadd_wch(int *y*, int *x*, const cchar_t **wch*);

int mvadd_wchnstr(int *y*, int *x*, const cchar_t **wchstr*, int *n*);

int mvadd_wchstr(int *y*, int *x*, const cchar_t **wchstr*);

int mvchgat(int *y*, int *x*, int *n*, attr_t *attr*, short *color*,
    const void **opts*);

int mvcur(int *oldrow*, int *oldcol*, int *newrow*, int *newcol*);

int mvdelch(int *y*, int *x*);

int mvderwin(WINDOW **win*, int *par_y*, int *par_x*);

int mvgetch(int *y*, int *x*);

int mvgetnstr(int *y*, int *x*, char **str*, int *n*);

int mvgetn_wstr(int *y*, int *x*, wint_t **wstr*, int *n*);

int mvgetstr(int *y*, int *x*, char **str*);

int mvget_wch(int *y*, int *x*, wint_t **ch*);

int mvget_wstr(int *y*, int *x*, wint_t **wstr*);

int mvhline(int *y*, int *x*, chtype *ch*, int *n*);

int mvhline_set(int *y*, int *x*, const cchar_t **wch*, int *n*);

chtype mvinch(int *y*, int *x*);
int mvinchnstr(int *y*, int *x*, chtype **chstr*, int *n*);
int mvinchstr(int *y*, int *x*, chtype **chstr*);
int mvinnstr(int *y*, int *x*, char **str*, int *n*);
int mvinnwstr(int *y*, int *x*, wchar_t **wstr*, int *n*);
int mvinsch(int *y*, int *x*, chtype *ch*);
int mvinsnstr(int *y*, int *x*, const char **str*, int *n*);
int mvinsstr(int *y*, int *x*, const char **str*);
int mvinstr(int *y*, int *x*, char **str*);
int mvins_nwstr(int *y*, int *x*, const wchar_t **wstr*, int *n*);
int mvins_wch(int *y*, int *x*, const cchar_t **wch*);
int mvins_wstr(int *y*, int *x*, const wchar_t **wstr*);
int mvinwstr(int *y*, int *x*, wchar_t **wstr*);
int mvin_wch(int *y*, int *x*, cchar_t **wcval*);
int mvin_wchnstr(int *y*, int *x*, cchar_t **wchstr*, int *n*);
int mvin_wchstr(int *y*, int *x*, cchar_t **wchstr*);
int mvprintw(int *y*, int *x*, char **fmt*, ...);
int mvscanw(int *y*, int *x*, char **fmt*, ...);
int mvvline(int *y*, int *x*, chtype *ch*, int *n*);
int mvvline_set(int *y*, int *x*, const cchar_t **wch*, int *n*);
int mvwaddch(WINDOW **win*, int *y*, int *x*, const chtype *ch*);
int mvwaddchnstr(WINDOW **win*, int *y*, int *x*, const chtype **chstr*,
int *n*);
int mvwaddchstr(WINDOW **win*, int *y*, int *x*, const chtype **chstr*);
int mvwaddnstr(WINDOW **win*, int *y*, int *x*, const char **str*, int *n*);
int mvwaddnwstr(WINDOW **win*, int *y*, int *x*, const wchar_t **wstr*,
int *n*);
int mvwaddstr(WINDOW **win*, int *y*, int *x*, const char **str*);
int mvwaddwstr(WINDOW **win*, int *y*, int *x*, const wchar_t **wstr*);
int mvwadd_wch(WINDOW **win*, int *y*, int *x*, const cchar_t **wch*);
int mvwadd_wchnstr(WINDOW **win*, int *y*, int *x*, const cchar_t **wchstr*,
int *n*);
int mvwadd_wchstr(WINDOW **win*, int *y*, int *x*, const cchar_t **wchstr*);
int mvwchgat(WINDOW **win*, int *y*, int *x*, int *n*, attr_t *attr*,
short *color*, const void **opts*);
int mvwdelch(WINDOW **win*, int *y*, int *x*);
int mvwgetch(WINDOW **win*, int *y*, int *x*);
int mvwgetnstr(WINDOW **win*, int *y*, int *x*, char **str*, int *n*);
int mvwgetn_wstr(WINDOW **win*, int *y*, int *x*, wint_t **wstr*, int *n*);

```
int    mvwgetstr(WINDOW *win, int y, int x, char *str);
int    mvwget_wch(WINDOW *win, int y, int x, wint_t *ch);
int    mvwget_wstr(WINDOW *win, int y, int x, wint_t *wstr);
int    mvwhline(WINDOW *win, int y, int x, chtype ch, int n);
int    mvwhline_set(WINDOW *win, int y, int x, const cchar_t *wch,
              int n);
int    mvwin(WINDOW *win, int y, int x);
chtype mvwinch(WINDOW *win, int y, int x);
int    mvwinchnstr(WINDOW *win, int y, int x, chtype *chstr, int n);
int    mvwinchstr(WINDOW *win, int y, int x, chtype *chstr);
int    mvwinnstr(WINDOW *win, int y, int x, char *str, int n);
int    mvwinnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);
int    mvwinsch(WINDOW *win, int y, int x, chtype ch);
int    mvwinsnstr(WINDOW *win, int y, int x, const char *str, int n);
int    mvwinsstr(WINDOW *win, int y, int x, const char *str);
int    mvwinstr(WINDOW *win, int y, int x, char *str);
int    mvwins_nwstr(WINDOW *win, int y, int x, const wchar_t *wstr,
              int n);
int    mvwins_wch(WINDOW *win, int y, int x, const cchar_t *wch);
int    mvwins_wstr(WINDOW *win, int y, int x, const wchar_t *wstr);
int    mvwinwstr(WINDOW *win, int y, int x, wchar_t *wstr);
int    mvwin_wch(WINDOW *win, int y, int x, cchar_t *wcval);
int    mvwin_wchnstr(WINDOW *win, int y, int x, cchar_t *wchstr,
              int n);
int    mvwin_wchstr(WINDOW *win, int y, int x, cchar_t *wchstr);
int    mvwprintw(WINDOW *win, int y, int x, char *fmt, ...);
int    mvwscanw(WINDOW *win, int y, int x, char *fmt, ...);
int    mvwvline(WINDOW *win, int y, int x, chtype ch, int n);
int    mvwvline_set(WINDOW *win, int y, int x, const cchar_t *wch,
              int n);
int    napms(int ms);
WINDOW *newpad(int nlines, int ncols);
SCREEN *newterm(char *type, FILE *outfile, FILE *infile);
WINDOW *newwin(int nlines, int ncols, int begin_y, int begin_x);
int    nl(void);
int    nonl(void);
int    nocbreak(void);
int    nodelay(WINDOW *win, bool bf);
int    noecho(void);
```

void noqiflush(void);
int noraw(void);
int notimeout(WINDOW **win*, bool *bf*);
int overlay(const WINDOW **srcwin*, WINDOW **dstwin*);
int overwrite(const WINDOW **srcwin*, WINDOW **dstwin*);
int pair_content(short *pair*, short **f*, short **b*);
int PAIR_NUMBER(int *value*);
int pechochar(WINDOW **pad*, chtype **ch*);
int pecho_wchar(WINDOW **pad*, const cchar_t **wch*);
int pnoutrefresh(WINDOW **pad*, int *pminrow*, int *pmincol*, int *sminrow*,
    int *smincol*, int *smaxrow*, int *smaxcol*);
int prefresh(WINDOW **pad*, int *pminrow*, int *pmincol*, int *sminrow*,
    int *smincol*, int *smaxrow*, int *smaxcol*);
int printw(char **fmt*, ...);
int putp(const char **str*);
int putwin(WINDOW **win*, FILE **filep*);
void qiflush(void);
int raw(void);
int redrawwin(WINDOW **win*);
int refresh(void);
int resetty(void);
int reset_prog_mode(void);
int reset_shell_mode(void);
int ripoffline(int *line*, int (**init*)(WINDOW **win*, int *columns*));
int savetty(void);
int scanw(char **fmt*, ...);
int scr_dump(const char **filename*);
int scr_init(const char **filename*);
int scrl(int *n*);
int scroll(WINDOW **win*);
int scrollok(WINDOW **win*, bool *bf*);
int scr_restore(const char **filename*);
int scr_set(const char **filename*);
int setcchar(cchar_t **wcval*, const wchar_t **wch*,
    const attr_t *attrs*, short *color_pair*,
    const void **opts*);
int setscrreg(int *top*, int *bot*);
SCREEN *set_term(SCREEN **new*);
int slk_attroff(const chtype *attrs*);

int slk_attr_off(const attr_t *attrs*, void **opts*);
int slk_attron(const chtype *attrs*);
int slk_attr_on(const attr_t *attrs*, void **opts*);
int slk_attrset(const chtype *attrs*);
int slk_attr_set(const attr_t *attrs*, short *color_pair*,
    void **opts*);
int slk_clear(void);
int slk_init(int *fmt*);
char *slk_label(int *labnum*);
int slk_noutrefresh(void);
int slk_refresh(void);
int slk_restore(void);
int slk_set(int *labnum*, const char **label*, int *justify*);
int slk_touch(void);
int slk_wset(int *labnum*, const wchar_t **label*, int *justify*);
int standend(void);
int standout(void);
int start_color(void);
WINDOW *subpad(WINDOW **orig*, int *nlines*, int *ncols*, int *begin_y*,
    int *begin_x*);
WINDOW *subwin(WINDOW **orig*, int *nlines*, int *ncols*, int *begin_y*,
    int *begin_x*);
int syncok(WINDOW **win*, bool *bf*);
chtype termattrs(void);
attr_t term_attrs(void);
char *termname(void);
int tigetflag(char **capname*);
int tigetnum(char **capname*);
char *tigetstr(char **capname*);
void timeout(int *delay*);
int touchline(WINDOW **win*, int *start*, int *count*);
int touchwin(WINDOW **win*);
char *tparm(char **cap*, long *p1*, long *p2*, long *p3*, long *p4*,
    long *p5*, long *p6*, long *p7*, long *p8*, long *p9*);
int typeahead(int *fildes*);
int ungetch(int *ch*);
int unget_wch(const wchar_t *wch*);
int untouchwin(WINDOW **win*);
void use_env(bool *boolvalue*);

int vidattr(chtype *attr*);
int vid_attr(attr_t *attr*, short *color_pair*, void **opts*);
int vidputs(chtype *attr*, int (**putfunc*)(int));
int vid_puts(attr_t *attr*, short *color_pair*, void **opt*,
 int (**putwfunc*)(int));
int vline(chtype *ch*, int *n*);
int vline_set(const cchar_t **wch*, int *n*);
int vwprintw(WINDOW **win*, char **fmt*, void **varglist*);
int vw_printw(WINDOW **win*, char **fmt*, void **varglist*);
int vwscanw(WINDOW **win*, char **fmt*, void **varglist*);
int vw_scanw(WINDOW **win*, char **fmt*, void **varglist*);
int waddch(WINDOW **win*, const chtype *ch*);
int waddchnstr(WINDOW **win*, const chtype **chstr*, int *n*);
int waddchstr(WINDOW **win*, const chtype **chstr*);
int waddnstr(WINDOW **win*, const char **str*, int *n*);
int waddnwstr(WINDOW **win*, const wchar_t **wstr*, int *n*);
int waddstr(WINDOW **win*, const char **str*);
int waddwstr(WINDOW **win*, const wchar_t **wstr*);
int wadd_wch(WINDOW **win*, const cchar_t **wch*);
int wadd_wchnstr(WINDOW **win*, const cchar_t **wchstr*, int *n*);
int wadd_wchstr(WINDOW **win*, const cchar_t **wchstr*);
int wattroff(WINDOW **win*, int *attrs*);
int wattron(WINDOW **win*, int *attrs*);
int wattrset(WINDOW **win*, int *attrs*);
int wattr_get(WINDOW **win*, attr_t **attrs*, short **color_pair*, void **opts*);
int wattr_off(WINDOW **win*, attr_t *attrs*, void **opts*);
int wattr_on(WINDOW **win*, attr_t *attrs*, void **opts*);
int wattr_set(WINDOW **win*, attr_t *attrs*, void **opts*);
int wbkgd(WINDOW **win*, chtype *ch*);
void wbkgdset(WINDOW **win*, chtype *ch*);
int wbkgrnd(WINDOW **win*, const cchar_t **wch*);
void wbkgrndset(WINDOW **win*, const cchar_t **wch*);
int wborder(WINDOW **win*, chtype *ls*, chtype *rs*, chtype *ts*, chtype *bs*,
 chtype *tl*, chtype *tr*, chtype *bl*, chtype *br*);
int wborder_set(WINDOW **win*, const cchar_t **ls*, const cchar_t **rs*,
 const cchar_t **ts*, const cchar_t **bs*,
 const cchar_t **tl*, const cchar_t **tr*,
 const cchar_t **bl*, const cchar_t **br*);
int wchgat(WINDOW **win*, int *n*, attr_t *attr*, short *color*,

```
            const void *opts);
int    wclear(WINDOW *win);
int    wclrtobot(WINDOW *win);
int    wclrtoeol(WINDOW *win);
void   wcursyncup(WINDOW *win);
int    wdelch(WINDOW *win);
int    wdeleteln(WINDOW *win);
int    wechochar(WINDOW *win, const chtype ch);
int    wecho_wchar(WINDOW *win, const cchar_t *wch);
int    werase(WINDOW *win);
int    wgetbkgrnd(WINDOW *win, cchar_t *wch);
int    wgetch(WINDOW *win);
int    wgetnstr(WINDOW *win, char *str, int n);
int    wgetn_wstr(WINDOW *win, wint_t *wstr, int n);
int    wgetstr(WINDOW *win, char *str);
int    wget_wch(WINDOW *win, wint_t *ch);
int    wget_wstr(WINDOW *win, wint_t *wstr);
int    whline(WINDOW *win, chtype ch, int n);
int    whline_set(WINDOW *win, const cchar_t *wch, int n);
chtype winch(WINDOW *win);
int    winchnstr(WINDOW *win, chtype *chstr, int n);
int    winchstr(WINDOW *win, chtype *chstr);
int    winnstr(WINDOW *win, char *str, int n);
int    winnwstr(WINDOW *win, wchar_t *wstr, int n);
int    winsch(WINDOW *win, chtype ch);
int    winsdelln(WINDOW *win, int n);
int    winsertln(WINDOW *win);
int    winsnstr(WINDOW *win, const char *str, int n);
int    winsstr(WINDOW *win, const char *str);
int    winstr(WINDOW *win, char *str);
int    wins_nwstr(WINDOW *win, const wchar_t *wstr, int n);
int    wins_wch(WINDOW *win, const cchar_t *wch);
int    wins_wstr(WINDOW *win, const wchar_t *wstr);
int    winwstr(WINDOW *win, wchar_t *wstr);
int    win_wch(WINDOW *win, cchar_t *wcval);
int    win_wchnstr(WINDOW *win, cchar_t *wchstr, int n);
int    win_wchstr(WINDOW *win, cchar_t *wchstr);
int    wmove(WINDOW *win, int y, int x);
int    wnoutrefresh(WINDOW *win);
```

int wprintw(WINDOW **win*, char **fmt*, ...);

int wredrawln(WINDOW **win*, int *beg_line*, int *num_lines*);

int wrefresh(WINDOW **win*);

int wscanw(WINDOW **win*, char **fmt*, ...);

int wscrl(WINDOW **win*, int *n*);

int wsetscrreg(WINDOW **win*, int *top*, int *bot*);

int wstandend(WINDOW **win*);

int wstandout(WINDOW **win*);

void wsyncdown(WINDOW **win*);

void wsyncup(WINDOW **win*);

void wtimeout(WINDOW **win*, int *delay*);

int wtouchln(WINDOW **win*, int *y*, int *n*, int *changed*);

wchar_t *wunctrl(cchar_t **wc*);

int wvline(WINDOW **win*, chtype *ch*, int *n*);

int wvline_set(WINDOW **win*, const cchar_t **wch*, int *n*);

实际应用信息

要支持包括 **<curses.h>** 的历史记录应用程序和使用 **<varargs.h>**，下列使用 **va_list** 的接口被声明为具有第三种类型 **(void *)** 的参数： **vw_printw()**、 **vw_scanw()**、 **vwprintw()**、 **vwscanw()**。

另请参阅

curses_intro(3X)、 **<stdio.h>**、 **<term.h>**、 **<termios.h>**、 **<unctrl.h>**、 **<wchar.h>**。

“X/Open System Interfaces and Headers, Issue 4, Version 2”规范。

已过时

名称

dbc_max_pct、dbc_min_pct、bufcache_max_pct、bufpages、nbuf - 已过时的内核可调参数

说明

这些可调参数已过时并被删除。请不要对这些可调参数作任何更改。因为它们将不会对内核产生任何影响。

使用文件缓存可调参数 **filecache_max** 和 **filecache_min** （请参阅 *filecache_max*(5)）。

在以前版本中，可调参数 **dbc_min_pct** 和 **dbc_max_pct** 用于设置对动态缓冲区缓存的限制；可调参数 **bufcache_max_pct**、 **bufpages** 和 **nbuf** 用于调整需要静态缓存时的缓冲区缓存。

本版本的 HP-UX 改进了文件缓存技术和与缓存文件 I/O 数据相关的物理内存控制。用于控制文件缓存内存使用的可调参数的数量从五个减少到两个，并且之前的可用性问题也得到了解决。

更改限制

不应修改这些可调参数。尝试调整任何已过时的缓冲区缓存的可调参数 **bufcache_max_pct** 、 **bufpages** 、 **dbc_min_pct** 、 **dbc_max_pct** 或 **nbuf** 都将会导致错误。

使用可调参数 **filecache_max** 和 **filecache_min** 设置对文件缓存的限制。请注意，在任何给定的系统上，这两个新可调参数的最佳值并不一定等于较早系统中已过时的缓冲区缓存的可调参数的最佳值。在尝试修改新文件缓存的可调参数值之前，应首先确定新缺省值在系统上是否会具有可接受的性能。

作者

bufcache_max_pct 、 **bufpages** 、 **dbc_min_pct** 、 **dbc_max_pct** 和 **nbuf** 由 HP 开发。

另请参阅

filecache_max(5)。

名称

default_disk_ir - 在 SCSI 子系统中启用和禁用设备写入缓存的使用（已过时）

值

保证安全

0（禁用）

缺省值

0（禁用）

允许值

0（禁用） 或非零（启用）

建议值

0（禁用）

说明

注释：此可调参数已过时。缺省情况下，HP 不再允许在系统上为所有直接访问设备启用写入缓存。但通过 **scsictl** 命令仍可以为特定设备启用写入缓存。请参阅 *scsictl*(1M)。

此可调参数启用 (1) 或禁用 (0) SCSI 子系统的立即报告行为，也称为写入缓存启用 (WCE)。如果启用了立即报告，当数据进行缓存时，具有数据缓存的磁盘驱动器从 **write()** 系统调用（包括原始写入）返回，而不是在该数据写入介质之后返回。这样有时会提高写入性能，尤其是对顺序传输。

如果在设备将缓存数据写入到介质之前该设备发生电源故障或重启，则缓存的数据将丢失。由于这些风险，服务器上此参数的建议值为立即报告禁用 (0)。

尽管不是挂接命令的选项，但此可调参数会对文件系统和裸磁盘性能产生很大影响，并且当发生系统重置时，对于数据完整性将产生不利影响。此可调参数也影响延迟写入与通过文件系统写入的行为。

更改此可调参数的人员

任何用户。

更改限制

对此可调参数的更改将在首次打开设备时生效。

应该何时启用此可调参数选项

当第三方应用程序供应商建议时。对正常使用情况，HP 强烈建议用户不要启用该可调参数。

启用此可调参数的负面影响

由于启用此可调参数改变了 LVM 或 RAID 提供的保护，因此如果发生设备电源故障或重启，则文件系统损坏及数据丢失的风险很大。

应该何时禁用此可调参数选项

HP 建议用户总是禁用此可调参数。如果用户不想冒设备电力故障或重置造成的文件系统损坏和数据丢失的风险，则尤其要这样做。

禁用此可调参数的负面影响

当消除设备电力故障或重置造成的文件系统损坏及数据丢失的风险时，这可能造成磁盘写入（不是读取）访问的性能下降。

应该同时更改的其他可调参数

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。对于 HP-UX 11i v3 而言，此参数已过时。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

default_disk_ir 由 HP 开发。

另请参阅

write(2)、 scsi(7)。

名称

dirent.h - 目录流和目录条目的格式

概要

#include <sys/types.h>

#include <dirent.h>

说明

该头文件定义了 *directory*(3C) 中描述的目录流例行程序所使用的数据类型。

定义了下列数据类型：

DIR	该结构包含关于打开目录流的信息。
struct dirent	该结构可定义由 **readdir()** 函数返回的条目的格式（请参阅 *directory*(3C)）。

struct dirent 结构包括下列成员：

```
char d_name[MAXNAMLEN+1];  /* name of directory entry */
ino_t d_ino;               /* file serial number */
short d_namlen;            /* length of string in d_name */
short d_reclen;            /* length of this record */
```

常量 **MAXNAMLEN** 在 **<dirent.h>** 中定义。

请注意， **d_reclen** 条目内部用于表示从当前条目到下一个有效条目的偏移量。因此， **d_reclen** 不是当前条目的长度，但是当前记录的长度是当前条目的位置加上当前条目与下一个有效条目之间的未使用空间的和。有效的 **dirent** 条目之间的未使用空间是由于目录内容的更改（例如删除文件和其他目录）而生成的。

本文件还包含对执行目录操作的函数的外部声明（请参阅 *directory*(3C)）。

警告

对于 32 位应用程序， *d_ino* 字段可能会对使用 64 位值的文件系统发生溢出。此种情况下，最重要的字节会被截断但是不会显示错误，并且该值可能不唯一。

作者

<dirent.h> 由 AT&T 和 HP 联合开发。

另请参阅

directory(3C)。

符合的标准

<dirent.h>： AES、SVID2、SVID3、XPG2、XPG3、XPG4、FIPS 151-2、POSIX.1

名称

diskaudit_flush_interval - 确定刷新审核记录的时间间隔（以秒为单位）

值

保证安全

1

缺省值

5

允许值

1 - 100

建议值

3 - 6

说明

可调参数 **diskaudit_flush_interval** 控制连续两次刷新审核记录的定期间隔，这些审核记录在与内核线程绑定的内核内存中进行缓冲。该内核线程具有 64K 缓冲区大小，用于保存审核记录，最多可以提供 32 个这样的线程。如果缓冲区已完全填满，这些内核线程将自动刷新其缓冲区，但是，根据系统中的活动，该过程可能需要较长的时间。

设置此可调参数值时，应确保当线程的缓冲区大约填充到一半时，或者线程长时间处于空闲状态，但缓冲区中仍保留某些数据时，线程会持续清理其缓冲区。保持过小的可调参数值将使线程刷新的频率过快，并导致过多的小规模写入操作，因而会影响性能。另一方面，保持过大的值可能导致消耗较多的未刷新内存。

更改此可调参数的人员

具有适当权限的管理员可以更改 **diskaudit_flush_interval** 的值。

更改限制

无。此可调参数为动态的。

应该何时增加此可调参数的值

系统不处理与审核相关的许多活动时。

增加此值的负面影响

如果系统正在生成大量的记录，同时增加此值，这样可能会导致出现大量未刷新的内存，从而可能会降低系统速度。

应该何时降低此可调参数的值

系统生成大量的审核记录时。

降低此值的负面影响

如果系统没有正在生成大量的记录，同时该值被减小，这样做可能导致出现大量的小规模写入，实际上只写入到缓冲区缓存，并不立即写入到物理磁盘，随后才进行实际写入。这样同样可能会降低性能。

应该同时更改的其他可调参数值

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

diskaudit_flush_interval 由 HP 开发。

另请参阅

audit(5)。

名称

dld.sl - 动态加载程序

多线程应用信息

动态加载程序是线程安全的。

说明

/usr/lib/pa20_64/dld.sl 程序是 PA-RISC 64 位动态加载程序。 **/usr/lib/dld.sl** 程序是 PA-RISC 32 位动态加载程序。在使用共享库的程序中，在 PA-RISC 64 位系统上， **dld.sl** 在启动时由 exec 自动调用，在 PA-RISC 32 位系统上则通过启动文件 **crt0.o** 调用。 **crt0.o** 的相同副本保存在 **/opt/langtools/lib** 和 **/usr/ccs/lib** 目录中。动态加载程序自身为共享库，虽然它没有定义任何供用户程序使用的符号。

共享库

共享库是可执行文件，由 **ld** 的 **-b** 选项创建（请参阅 *ld*(1) ）。它们必须包含与位置无关的代码 (PIC)，后者在进程的地址空间中可以随处映射，并以最小重定位执行。 PIC 可以使用 PC和（或）链接表。缺省情况下，在 PA-RISC 64 位系统下它通过编译程序生成，在 PA-RISC 32 位系统上则通过指定 **+z/+Z** 选项生成。有关用汇编语言编写 PIC 的详细信息，请参阅 *ld*(1) 的 **+help** 选项或《HP-UX Linker and Libraries User's Guide》手册。

不完整的可执行文件

与一个或多个共享库链接的可执行程序，称为 “不完整的可执行文件”。

从对象文件和库创建可执行文件（ a.out ）时，链接程序不从共享库内复制文本（代码）或数据到输出文件。但是，动态加载程序在运行时会将库映射到进程的地址空间。链接程序将所有程序引用绑定到共享库例行程序，将数据绑定到链接表中的条目，并且一旦将库映射后，会依赖动态加载程序填充该链接表条目。链接表用作函数调用的跳转表。

当前支持线程本地存储。动态加载程序统计每个共享库的线程本地存储大小和程序的线程本地存储大小。当所有库加载到 PA-RISC 32 位系统时，动态加载程序调用线程例行程序设置线程指针并为初始线程分配空间，或调用 **mmap()** 和 **lwp_setprivate()** 分配空间并设置线程指针。在 PA-RISC 64 位系统上，动态加载程序在系统库 **libc** 中调用执行线程初始化、分配初始线程和设置线程指针的初始设置。

在先前的 PA-RISC 32 位版本中，程序引用的共享库数据项目被复制到程序可执行文件中，以便静态解析数据引用。从 700/800 系列 10.0 版本开始，对 **a.out** 中共享库数据的引用被包含在链接表中并在运行时解析。

加载

不完整的可执行文件包含在链接时间进行搜索的共享库路径名列表。在运行时，动态加载程序将与程序链接的所有共享库连接到进程。动态加载程序尝试从在链接时在其中找到库的同一目录加载每个库。可以通过指定动态路径列表更改共享库运行搜索路径。请参阅“PA-RISC 32 位动态路径列表”和（或）“PA-RISC 64 位动态路径列表”。

库的文本段在所有使用该库的进程中共享。数据和 bss 段在逐页基础上共享。当进程初次访问（读或写）数据或 bss 页面时，为该进程生成该页面的副本。

PARISC 32 位动态路径列表

有两种方式指定动态路径列表：

- 通过使用 **ld** 的 **+b** 路径列表选项，在可执行文件中存储目录路径列表。
- 通过链接可执行文件和 **ld** 的选项 **+s** ，允许可执行文件在运行时使用 **SHLIB_PATH** 环境变量定义的路径列表。

路径列表是以冒号 (:) 分隔的一个或多个路径名称的列表。动态路径列表仅对使用 **ld** 的 **-l** 或 **-l:** 选项指定的库工作。但也可对使用 **chatr** （请参阅 *chatr*(1) ）的 **-l** 选项由完整路径名指定的库启用。如果同时使用了 **+s** 和 **+b** ，则它们在命令行上的相对顺序指示在兼容模式下将首先搜索哪个路径列表。有关详细信息，请参阅 *ld*(1) 的 **+help** 选项或《HP-UX Linker and Libraries User's Guide》。

PA-RISC 64 位动态路径列表

对于标准模式库（使用 **ld +std** 构建或与之链接的库），动态加载程序将使用动态路径搜索以查找名称出现在程序共享库列表中的共享库，或者无嵌入 **/** 字符的已加载共享库。对这些标准模式库或可执行文件，缺省情况启用动态路径搜索。如果 **ld +noenvvar** 已经指定，则动态加载程序不查看任何动态路径环境变量以查找相关共享库。这将动态路径搜索限制为 **rpath** 的值以及缺省目录 **/usr/lib/pa20_64** 和 **/usr/ccs/lib/pa20_64** 。

对于兼容模式库（使用 **ld +compat**构建或与之链接的库），如果这些库与 **-l** 或 **-l:** 链接，并指定了下列之一，动态加载程序将仅进行动态路径搜索。

- **ld +s**
- **ld +b**
- **chatr +s enable**

有多种方法指定动态路径列表：

- 通过使用 **ld** 的 **+b** *path_list* 选项在可执行文件中存储目录路径列表。
- 通过不指定 **ld +b** ，并让链接程序将 **rpath** 值设置为 **ld -L** 路径列表的连接，其后为环境变量 **LPATH** 的值，然后为缺省目录 **/usr/lib/pa20_64** 和 **/usr/ccs/lib/pa20_64** 。此项仅适用于标准模式共享库。
- 通过在环境变量 **LD_LIBRARY_PATH** 和（或） **SHLIB_PATH** 中存储目录路径列表。对于兼容模式共享库和可执行文件，目录路径列表应该仅放置在 **SHLIB_PATH** 环境变量中。

路径列表是以冒号 **(:)** 分隔的一个或多个路径名称的列表。动态路径列表仅对使用 **ld** 的 **-l** 或 **-l:** 选项指定的库工作。但也可对使用 **chatr** （请参阅 *chatr*(1) ）的 **-l** 选项由完整路径名指定的库启用。如果同时使用了 **+s** 和 **+b** ，则它们在命令行上的相对顺序将指示在兼容模式下首先搜索哪个路径列表。有关详细信息，请参阅 *ld*(1) 的 **+help** 选项或《HP-UX Linker and Libraries User's Guide》。

动态加载程序在确定要使用哪些动态路径列表时将使用下列规则：

- 如果 **ld +noenvvar** 已经指定并且 **ld +b** 和 **ld +compat** 未指定，所能执行的动态路径搜索仅为查看 **rpath** 中的路径列表，其后为缺省目录 **/usr/lib/pa20_64** 和 **/usr/ccs/lib/pa20_64** 。

- 如果 **ld +compat** 已经指定并且 **ld +b** 和 **ld +s** 未指定，则不对共享库进行动态路径搜索。
- 如果 **ld +compat** 和 **ld +b** 选项未指定，则搜索 **LD_LIBRARY_PATH** 环境变量中的路径列表，其后是 **SHLIB_PATH** 环境变量中的路径列表，然后是 **rpath** 中的路径列表，之后为缺省目录 **/usr/lib/pa20_64** 和 **/usr/ccs/lib/pa20_64** 。
- 如果 **ld +compat** 、 **ld +b** 和 **ld +s** 已指定，则使用 **ld +b** 和 **ld +s** 相对顺序确定动态加载程序是否将使用 **rpath** 中 **SHLIB_PATH** 之前的路径列表，然后为共享库列表中指定的库。如果首先指定了 **ld +b** ，则先使用 **rpath** 中的路径列表。

在搜索相关共享库的时候规则会有细微更改。

- 对于标准模式库，首先搜索 **LD_LIBRARY_PATH** 环境变量中的路径列表，然后是 **SHLIB_PATH** 环境变量中的路径列表，接着是父共享库的 **rpath** 中的值，最后为缺省目录 **/usr/lib/pa20_64** 和 **/usr/ccs/lib/pa20_64** 。父共享库的祖先可能包含 **rpath** 中的路径列表，但这在搜索该父共享库的相关共享库时被忽略。仅使用父共享库的 **rpath** 。如果已为程序加载具有相同基名（不带路径的库名）的库，则该库用于解析相关共享库。
- 对于兼容模式库，搜索与父共享库相同，不同的是 **rpath** 可以从父共享库传递给子库各相关项的相关共享库等。

绑定

动态加载程序还解析可执行文件和库之间的符号引用。缺省情况下，通过链接表，函数调用在初次引用时被捕获并绑定。数据符号的引用和其他绝对地址引用不能被捕获。它们与函数调用的第一个解析进行绑定，该函数可能引用对象。

如果使用 **ld** 的 **-B immediate** 选项，加载程序在启动时绑定所有必需的引用。该项增加程序启动时的开销，但可确保以后无需更多的绑定操作。因而可以获得较好的实时响应，并可消除随后未解析外部引用带来的异常中止风险。

fastbind 工具可用于加快使用共享库（不完整的可执行文件）的程序的启动速度。 **fastbind** 工具分析共享库例行程序和用于绑定符号的数据，并且在可执行文件中存储该信息。动态加载程序将通知此信息可用，并使用此快速绑定信息绑定符号而不是标准搜索方法。有关详细信息，请参考 *fastbind*(1) 和 *ld*(1) 的 **+help** 选项，或《HP-UX Linker and Libraries User's Guide》手册。

横向优先搜索

仅在 PA-RISC 64 位系统上可用。缺省情况下，动态加载程序在绑定符号时执行横向优先搜索。如果不完整可执行文件与 **+compat** 链接或如果 **shl_load()** 正被执行，则使用纵向优先搜索。横向优先搜索指定动态加载程序从不完整可执行文件开始搜索，然后搜索所有已加载的共享库来查找符号，顺序从左向右，直至找到符号。例如，先搜索不完整可执行文件，然后搜索在其共享库列表中的全部库。然后搜索共享库列表中的第一个库的相关共享库，然后是列表中的第二个库的相关共享库，依此类推。

纵向优先搜索

这是 PA-RISC 32 位系统上唯一的搜索行为；在 PA-RISC 64 位系统上，如果执行 **shl_load()** 或不完整可执行文件与 **+compat** 链接时将使用此行为。动态加载程序搜索不完整可执行文件，其后是其共享库列表中的第一个库。然

后搜索此库的第一个相关库，接着为此相关库的第一个相关库，依此类推。如果没有更多的相关库，则搜索同属及其相关库，直至搜索了程序的共享库列表中的第二个库为止，然后是该库的第一个相关库，依此类推。

版本控制

由于共享库的代码在运行时从单独共享库文件中映射，对共享库的修改可能更改现有可执行文件的行为。在某些情况下，可能导致程序的错误操作。有两种版本控制方法解决此问题。

内部库版本

仅在 PA-RISC 32 位系统上可用。每当对库接口进行了不兼容的更改时，受影响模块或模块的版本都将包含在库里。指示更改发生日期（月/年）的标记在 C 中通过 **HP_SHLIB_VERSION** 存储于新模块，或在 Fortran 和 Pascal 中通过编译程序指令 **SHLIB_VERSION** 存储。该日期应用于模块内定义的所有符号。提供最近的不兼容更改日期的高水印记录在共享库中，与程序相链接的每个库的高水印记录在不完整可执行文件中。运行时，动态加载程序检查每个库的高水印并仅在库的更新时间至少与链接时记录的高水印时间相同时加载该库。在绑定符号引用时，加载程序选择符号最近版本，该版本不迟于链接时记录的高水印时间。这两项检查帮助确定运行时使用的每个库接口的版本与链接时的预期相同。内部库版本在未来版本中可能删除。

库级别版本

用户控制库版本的第二个方法是使用新命名约定 **libname.n** ，其中 **n** 为数字，它随着库的新版本的发布递增。使用新命名方案时，用户在构建共享库时必须使用 **ld** 的 **+h** *internal_name* 选项为共享库指定内部名称。此内部名称记录在与此共享库链接的每个不完整可执行文件或共享库中。

运行时，加载程序查看记录在不完整可执行文件或共享库中的共享库列表。对列表中不是内部名称的每个库，动态加载程序将查找库的 **.0** 版本（例如 **libname.0** ）以加载。如果未找到该版本，则它将查找记录在列表中的库名称。

PA-RISC 32 位显式加载和绑定

上文介绍的动态加载程序的任务都是自动执行的，虽然通过相应的 **ld** 选项可以对它们进行某些控制。动态加载程序也可以通过程序访问。保留的符号 **__dld_loc** 在 **crt0.o** 中定义，它指向动态加载程序中的跳转表。*shl_load*(3X) 下描述的例行程序提供了可移植接口，该接口允许编程人员在运行时将共享库显式连接到进程、及计算共享库内定义的符号地址并在完成时分离库。

PA-RISC 64 位显式加载和绑定

上文介绍的动态加载程序的任务都是自动执行的，虽然通过相应的 **ld** 选项可以对它们进行某些控制。动态加载程序也可以通过程序访问。 *shl_load*(3X) 、 *dlclose*(3C) 、 *dlerror*(3C) 、 *dlget*(3C) 、 *dlmodinfo*(3C) 、 *dlopen*(3C) 和 *dlsym*(3C) 下说明的例行程序提供可移植接口，此接口允许程序员在运行时将共享库显式添加到进程、及计算共享库中定义的符号的地址和在完成时分离库。

全局符号表

全局符号表机制作为性能提升选项而设计。启用此机制会创建全局符号表以加速符号查找，这是通过在查找符号时减少扫描全部已加载库的需要来实现的。此机制允许动态加载程序从所有已加载库中只搜索一个包含符号信息的表。这在应用程序有大量的共享库时尤为有效。缺省情况下将关闭此机制。

全局符号表使用散列表来实现。此机制下，在加载库时（不论是隐式加载或使用 **dlopen()** 或 **shl_load()** ），此机制散列库的输出符号并将它们放入此表。当库被卸载时，该机制在表中查找库的输出符号并将它们删除。

散列表不包含由 **shl_definesym()** 定义的符号条目。因而用户定义的符号必须单独处理。

启用此机制使 **dld** 使用更多内存并影响 **dlopen()** 、 **dlclose()** 、 **shl_load()** 和 **shl_unload()** API 调用的性能。

全局符号表机制可以强制动态加载程序 (dld.sl) 执行大量的散列操作以定位符号。执行此散列函数需要消耗大量时间，尤其在符号名称非常长的时候（C++ 程序）。要加速加载程序，可以使用 **+gst** 选项卸载链接程序的计算散列值。这使得链接程序计算从链接行上列出的库导出的全部符号的散列值，并在可执行文件中存储这些散列值。运行时，加载程序在内存中构建全局符号表并在每个库加载时从可执行文件读取存储的散列值。如果在链接时不指定 **+gst** ，可以使用 *chatr*(1) 命令的 **+gst** *flag* 选项启用全局符号表机制，这使加载程序在运行时创建表并计算散列值。

使用 GST 选项（通过 **ld** 和 **chatr** 命令）， **+gst** 、 **+gstbuckets** （仅用于 PA-RISC 32 位）、 **+gstsize** 、 **+nodynhash** （仅用于 PA-RISC 64 位系统）和 **+plabel_cache** （仅用于 PA-RISC 32 位系统）控制全局符号表散列机制的行为。可以使用 **+gstsize** 和 **+nodynhash** linker/chatr 选项控制全局符号表散列机制的行为。使用 GST 选项可以调节散列表的 大小和每个条目容器的 数目，以达到性能和内存使用的平衡。要获得最高性能，调节表的大小为某个平均链长度。要通过降低性能获得最大内存使用，可调节表的大小以使空条目的数目最小。通常，使用质数作为表的大小。机制提供的表大小的缺省值为 1103，容器数的缺省值为 3。

要获得散列表性能的静态信息，请设置环境变量 **_HP_DLDOPTS** 以包含 **-symtab_stat** 选项。此选项为每个包含下列信息的库提供一条消息：

- 操作（加载/卸载）
- 库名称
- 导出符号数
- 表中无存储符号的条目的数目
- 非零链的平均长度
- 计算散列表的性能
- 散列表使用内存数量

有关 **+gst** 、 **+gstbuckets** 、 **+gstsize** 、 **+nodynhash** 和 **+plabel_cache** 选项的信息，请参阅 *ld*(1) 和 *chatr*(1) 。

LD_PRELOAD 环境变量

注释： **LD_PRELOAD** 功能在 seteuid/setegid 程序中禁用，如 **passwd** 。有关详细信息，请参阅 *ld*(1) 。此功能对完全绑定静态可执行文件不可用。

LD_PRELOAD 环境变量允许用户在程序启动时加载附加共享库。 **LD_PRELOAD** 提供动态加载程序可以解释的以冒号分隔或空格分隔的共享库列表。动态加载程序 **dld.sl** 加载指定共享库，就像在其他相关库之前，此程序已经在 **LD_PRELOAD** 中与此共享库显式链接。

启动时，动态加载程序隐式加载 **LD_PRELOAD** 环境中指定的一个或多个库（如果有）。它将使用相同的加载顺序和符号解析顺序，就像在创建可执行文件时，此库已经作为链接行中的第一个库被显式链接。例如，给定一个使用以下链接行构建的可执行文件：

$ ld ... lib2.sl lib3.sl lib4.sl

如果 **LD_PRELOAD="/var/tmp/lib1.sl"** ，动态加载程序使用相同加载顺序和符号解析顺序，就像 **lib1.sl** 已经被指定为链接行的第一个库：

$ ld ... /var/tmp/lib1.sl lib2.sl lib3.sl lib4.sl

典型的命令行使用（用 **/bin/sh** ），其中 **LD_PRELOAD** 定义如下：

$ LD_PRELOAD=mysl.sl *application*

动态加载程序根据 **$PATH** 搜索 *application* ，但根据 **SHLIB_PATH** 和（或） **LD_LIBRARY_PATH** ，和（或）嵌入路径（如果已启用）搜索 **mysl.sl** 。

注释：因为动态加载程序在运行任何可执行文件（除 seteuid/setegid 程序外）时检查 **LD_PRELOAD** 环境变量，如果要导出 **LD_PRELOAD** ，则必须在运行完可执行文件后取消其设置，或者在上文列举的命令或脚本中运行可执行文件。

可以使用 **LD_PRELOAD** 环境变量加载共享库，该共享库包含线程本地存储，以避免在动态加载库时出现以下错误：

/usr/lib/dld.sl: Can't shl_load() a library containing Thread Local
Storage: /usr/lib/libcps.1

PA-RISC 32 位程序和相同的 PA-RISC 64 位程序的加载顺序和符号解析顺序会不同，这是因为动态加载程序在 PA-Risc 32 位模式下使用纵向优先搜索顺序，而在 RISC 64 位模式下使用横向优先搜索顺序。有关详细信息，请参阅 *ld*(1) 的 **+help** 选项中的"Symbol Searching and Dependent Libraries"，或者《HP-UX Linker and Libraries User's Guide》。

动态加载程序使用 **LD_PRELOAD** 环境变量，即使在链接行中使用 **+noenvvar** 也是如此。这就确保了 **LD_PRELOAD** 的启用，即便是在 **+compat** 链接中。 **LD_PRELOAD** 变量总是启用的，除非是在 setuid 和 setgid 程序中。

注释：如果归档库已经与应用程序链接，而用户试图使用 **LD_PRELOAD** 加载该库的共享版本，将会出现问题。

可能遇到的问题有：

- 如果库有初始设置/终结程序，则会被调用两次。
- 可能以相同数据符号的两个不同副本结束，这将导致错误的行为。

可以指定多个库为 **LD_PRELOAD** 环境变量的一部分。使用空格或冒号分隔库，如 **LD_LIBRARY_PATH**. （不将多字节支持作为解析 **LD_PRELOAD** 库列表的一部分提供）。可以使用绝对路径或相对路径指定 **LD_PRELOAD** 库。 **LD_PRELOAD** 库可仅由库名称组成，在这种情况下，动态加载程序使用环境变量 **LD_LIBRARY_PATH** 和（或） **SHLIB_PATH** 中的路径列表或嵌入路径列表（如果启用）来搜索库。

如果动态加载程序不能找到 **LD_PRELOAD** 指定的库，则不发出错误或警告消息。但是，如果它没有找到 **LD_PRELOAD** 库的相关库，动态加载程序发出相同的错误消息，就像是在链接行中指定了 **LD_PRELOAD** 库一样。

LD_PRELOAD_ONCE 环境变量

LD_PRELOAD_ONCE 功能与 **LD_PRELOAD** 相似，不同的是动态加载程序 **dld.sl** 在读取 **LD_PRELOAD_ONCE** 之后取消其设置，因此任何被当前应用程序调用的应用程序没有 **LD_PRELOAD_ONCE** 设置。此功能可用于当前应用程序需要预先加载某个库，而子应用程序会受到此预先加载库的反面影响的情况（例如，**ksh** 在 **LD_PRELOAD** 包含 **/usr/lib/libpthread.1** 时会意外终止）。**LD_PRELOAD_ONCE** 指定的库比 **LD_PRELOAD** 指定的库先被加载。符号解析的效果是 **LD_PRELOAD_ONCE** 指定的库中的符号具有比 **LD_PRELOAD** 指定的库中的符号更高的优先级。

诊断信息

如果由于任何原因动态加载程序不存在或不能被进程调用，则输出错误消息到标准错误并且进程以非零退出代码终止。

这些错误可归结为两个基本类别：连接共享库时的错误和绑定符号时的错误。前者仅在进程启动时出现，但后者可在执行进程的任何时间出现，除非在 **ld .** 中使用了 **-B immediate** 选项。在连接共享库时可能出现的错误包括库不存在、库不可执行、库损坏、高水印过低或库的地址空间不足。绑定符号时可能出现的错误包括符号未找到（未解析的外部引用）或库损坏。

使用动态加载程序的显式加载设备时，这些不是致命错误类型。有关详细信息，请查阅 *shl_load*(3X)、*dlclose*(3C)、*dlget*(3C)、*dlgetname*(3C)、*dlmodinfo*(3C)、*dlopen*(3C) 和 *dlsym*(3C)。在 PA-RISC 64 位系统上使用 **dlerror()** 例行程序查看错误消息。该例行程序将输出动态加载程序记录的最近错误消息。

警告

动态加载程序的启动开销非常大，即使使用延时绑定也是如此，在以启动开销为主的进程（例如简单“hello world”程序）中可能导致严重的性能降低。此外，独立位置代码一般比普通代码慢，所以在共享库中程序的性能可能会受到 PIC 出现的反面影响。但在多数情况下，可执行文件的低磁盘空间占用和低内存需求的优点超过了这些顾虑。

很少出现程序使用共享库与相对归档库时会有不同。这种情况主要发生在依赖于编译程序、汇编程序和链接程序的无正式文件和不支持功能时。有关详细信息，请参阅 *ld*(1) 的 **+help** 选项或《HP-UX Linker and Libraries User's Guide》。

库开发人员对版本控制负有全责，并必须彻底标识对库接口进行的不兼容更改。否则，在接下来的版本中可能出现无法预知的程序故障。如果库开发人员没有正确处理版本控制，则应用程序用户无能为力。应用程序开发人员通常可以通过修改源代码以使用新接口来解决问题，然后对新库重新编译和重新链接。

缺省情况下，动态加载程序不报告大多数警告。

在 PA-RISC 32 位系统上，如果希望查看全部消息，将环境变量 **_HP_DLDOPTS** 设置为包含一个或多个选项。支持下列选项：

-warnings 显示附加动态加载程序警告消息。其中一些问题包括：

- 同名但不同类型的符号，如 CODE 和 DATA。有关此警告的详细信息，请参阅 *ld*(1) 中的警告一节。

- 使用 *shl_load*(3X) 描述的某些标志或例行程序。

-fbverbose 请参阅 *fastbind*(1) 。

-nofastbind 请参阅 *fastbind*(1) 。

在 PA-RISC 64 位系统上，如果希望查看所有错误消息，请将环境变量 **DLD_VERBOSE_ERR** 设置为 True。

作者

/usr/lib/dld.sl 和 **/usr/lib/pa20_64/dld.sl** 共享库由 HP 开发。

另请参阅

系统工具

aCC(1)	调用 HP-UX aC++ 编译程序
as(1)	将汇编代码转换为机器代码
CC(1)	调用 HP-UX C++ 编译程序
cc(1)	调用 HP-UX C 编译程序
chatr(1)	更改程序的内部属性
f77(1)	调用 HP-UX FORTRAN 编译程序
f90(1)	调用 HP-UX Fortran 90 编译程序
fastbind(1)	调用 fastbind 工具
ld(1)	调用链接编辑器
pc(1)	调用 HP-UX Pascal 编译程序

其他信息

a.out(4)	汇编程序、编译程序和链接程序输出
dlclose(3C)	卸载先前由 **dlopen()** 加载的共享库
dlerror(3C)	输出由 dld 记录的最后错误信息
dlget(3C)	返回有关已加载模块的信息
dlgetname(3C)	返回包含加载模块的存储名称
dlmodinfo(3C)	返回有关已加载模块的信息
dlopen(3C)	加载共享库
dlsym(3C)	获取共享库中符号的地址
shl_load(3X)	加载/卸载共享库

文本和教程

《HP-UX Linker and Libraries User's Guide》
(请参阅 *ld*(1) 的 **+help** 选项)

《HP-UX Linker and Libraries User's Guide》
(有关订购信息，请参阅 *manuals*(5))

名称

dld.so - 动态加载程序

多线程应用信息

动态加载程序是线程安全的。

说明

/usr/lib/hpux64/dld.so 共享库是 64 位动态加载程序。 **/usr/lib/hpux32/dld.so** 共享库是 32 位动态加载程序。在使用共享库的程序中，启动时将自动调用 **dld.so** 。 **/usr/ccs/lib/hpux64/crt0.o** 是 64 位运行时启动文件。 **/usr/ccs/lib/hpux32/crt0.o** 是 32 位运行时启动文件。 **/usr/lib/hpux32/uld.so** 和 **/usr/lib/hpux64/uld.so** 共享库是 32 位和 64 位微加载程序。在使用共享库的程序中， *exec*(2) 在启动时将自动调用微加载程序。微加载程序唯一职责是向内存加载动态加载程序、 **dld.so** ，用于程序的执行。微加载程序 (**uld.so**) 和动态加载程序 (**dld.so**) 自身是共享库，即使它们没有定义供用户程序使用的符号也是如此。

共享库

共享库是使用 **ld** 的 **-b** 选项创建的可执行文件（请参阅 *ld*(1) ）。它们必须包含与位置无关的代码 (PIC)，该代码可在进程的地址空间中随处映射，并可通过最小重定位进行执行。 PIC 可以使用 PC 相关的寻址模式和（或）链接表。缺省情况下，HP 编译程序将生成 PIC。

不完整的可执行文件

与一个或多个共享库链接的可执行程序称为 “不完整的可执行文件”。

从对象和库中创建可执行文件 (**a.out**) 时，链接程序不会将文本（代码）或数据从共享库复制到输出文件。但是，动态加载程序会在运行时将库映射到进程的地址空间。链接程序将所有对共享库例行程序和数据的引用绑定到链接表中的条目，并在映射库之后，依靠动态加载程序来填写链接表条目。该链接表用作进行函数调用时的跳转表。

线程本地存储

当前支持两种线程本地存储模式：由编译程序选项 **+tls=static/dynamic** 控制的静态模式和动态模式。缺省为 **+tls=dynamic** 。使用动态模式构建的共享库可以用 *dlopen*(3C) 和 *shl_load*(3X) API 加载。尝试使用 *dlopen*(3C) 或 *shl_load*(3X) API 加载用静态模式构建的共享库将导致下列错误：

/usr/lib/hpux[32|64]/dld.so: Can’t shl_load() a library containing
Thread Local Storage: /usr/lib/hpux[32|64]/libcps.so.1

动态加载程序可统计每个共享库的线程本地存储大小和程序的线程本地存储大小。加载所有库之后，动态加载程序将调用系统库 **libc** 中的初始设置，该项可进行线程初始化、初始线程分配并设置线程指针。

加载

不完整的可执行文件包含链接时进行搜索的共享库路径名列表。在运行时，动态加载程序可连接到将所有共享库与程序链接的进程。动态加载程序将尝试在同一目录下加载链接时找到的每个库。可以通过指定动态路径列表更改共享库运行搜索路径（有关 PA-RISC 32 位兼容模式信息，请参阅动态路径列表）。

库的文本段将在所有使用该库的所有进程中共享。数据和 bss（未初始化的数据）段将逐页加载。当进程初次访问（读取或写入）数据或 bss 页面时，将生成该进程的该页面的副本。

动态路径列表

对于缺省模式库，动态加载程序使用动态路径搜索以查找其名称出现在程序的库列表中的共享库，或者用无嵌套的 **/** 字符加载的共享库。对于这些库或可执行文件，缺省情况下将启用动态路径搜索。如果已指定 **ld +noenvvar**，则动态加载程序不会查看任何动态路径环境变量来查找相关共享库。这会将动态路径搜索限制为 **rpath** （ **ld +b** 命令设置的运行时路径或嵌套路径）的值和 64 位库的缺省目录 **/usr/lib/hpux64** 或 32 位库的缺省目录 **/usr/lib/hpux32** 。

对于 PA-RISC 32 位兼容模式库（使用 **ld +compat** 构建或与之链接的库），如果这些库与 **-l** 或 **-l:** 链接，并且指定了下列之一，则动态加载程序仅对它们进行动态路径搜索。

- **ld +s**
- **ld +b**
- **chatr +s enable**

有多种途径指定动态路径列表:

- 通过使用 **ld .** 的 **+b** *path_list* 选项，在可执行文件（在 **rpath** 中）内存储目录路径列表。
- 通过 不指定 **ld +b** ，并且允许链接程序将 **rpath** 值设置为下列连接形式： **ld -L** *path_list* 后接环境变量 **LPATH** 的值，再后接 64 位缺省目录 **/usr/lib/hpux64** ，或 32 位缺省目录 **/usr/lib/hpux32** 。
- 通过在环境变量 **LD_LIBRARY_PATH** 和（或） **SHLIB_PATH** 中存储目录路径列表。 **LD_LIBRARY_PATH** 和 **SHLIB_PATH** 可以包含 32 位和 64 位库的路径；dld.so 仅加载相应的库。对于兼容模式共享库和可执行文件，环境列表应仅置入 **SHLIB_PATH** 环境变量中。

路径列表是以冒号 **(:)** 分隔的一个或多个路径名称列表。动态路径列表仅适用于用 **ld** 的 **-l** 或 **-l:** 选项指定的库。但是，使用 **chatr** 的 **-l** 选项，可将其用于完整路径名指定的库（请参阅 *chatr*(1) ）。如果同时使用 **+s** 和 **+b** ，则它们在命令行上的相对顺序将指示兼容模式下首先搜索的路径列表。有关详细信息，请参阅 *ld*(1) 的 **+help** 选项或《HP-UX Linker and Libraries User's Guide》。

动态加载程序确定要使用的动态路径列表时将使用下列规则:

- 如果指定了 **ld +noenvvar** ，而 没有指定 **ld +b** 和 **ld +compat** ，则唯一可以进行的动态路径搜索是查看 **rpath** 中的路径列表，其后接 64 位缺省目录 **/usr/lib/hpux64** ，或 32 位缺省目录 **/usr/lib/hpux32** 。
- 如果已指定 **ld +compat** ，而 没有指定 **ld +b** 和 **ld +s** ，则不对共享库进行动态路径搜索。使用库的链接时间位置（记录的路径）。
- 如果 没有指定 **ld +compat** 和 **ld +b** 选项，则搜索 **LD_LIBRARY_PATH** 环境变量中的路径列表，后接 **SHLIB_PATH** 环境变量中的路径列表，后接 **rpath** ，再后接 64 位缺省目录 **/usr/lib/hpux64** ，或 32 位缺省目录 **/usr/lib/hpux32** 。
- 如果已指定 **ld +compat** 、 **ld +b** 和 **ld +s** ，则将使用 **ld +b** 和 **ld +s** 的相对顺序确定动态加载程序是否应在 **SHLIB_PATH** 之前使用 **rpath** 的路径列表，后接库列表中指定的库。如果首先指定 **ld +b** ，则首先使用 **rpath** 中的路径列表。

搜索相关共享库的时候规则会稍为更改。

- 对于缺省模式库，首先搜索 **LD_LIBRARY_PATH** 环境变量中的路径列表，后接 **SHLIB_PATH** 环境变量中的路径列表，后接父共享库的 **rpath** 中的值，再后接 64 位缺省目录 **/usr/lib/hpux64** ，或者 32 位缺省目录 **/usr/lib/hpux32** 。父共享库的祖先可能在 **rpath** 中包含路径列表，但在搜索该父共享库的相关共享库时将其忽略了。仅使用父共享库的 **rpath** 。
- 对于兼容模式库，其搜索方式与父共享库的搜索方式相同，但是 **rpath** 可以从父共享库传递到子共享库及子库的相关共享库，依此类推。

绑定

动态加载程序还可解析可执行文件和库之间的符号引用。缺省情况下，初次引用时链接表和绑定会捕获函数调用。无法捕获数据符号的引用和其他绝对地址引用。它们将在首次解析可能引用对象的函数调用时绑定。

如果使用 **ld** 的 **-B immediate** 选项，则加载程序在运行时将绑定所有必需的引用。这会增加程序启动时的成本，但可确保以后无需进行更多绑定操作。因此可以产生更好的实时响应，并可消除未解析外部引用而导致的随后异常中止的风险。

fastbind 工具可用于减少使用共享库（不完整的可执行文件）的程序的启动时间。 **fastbind** 工具可分析用于绑定符号的共享库例行程序和使用的数据，并将该信息存储在执行文件中。动态加载程序通知此程序可用，并使用此快速绑定信息，而不是标准搜索方法绑定符号。有关详细信息，请参考 *fastbind*(1) 和 *ld*(1) 的 **+help** 选项或《HP-UX Linker and Libraries User's Guide》。

横向优先搜索

缺省情况下，动态加载程序在绑定符号时将执行横向优先搜索。如果将不完整的可执行文件与 **+compat** 链接或正在执行 **shl_load()** ，则使用纵向优先搜索（请参阅纵向优先搜索）。横向优先搜索指定动态加载程序查找以不完整的可执行文件开头，后接所有加载的共享库的符号（顺序从左至右），直至找到该符号为止。例如，在不完整可执行文件的加载列表中的所有库之后对其进行搜索。然后搜索库加载列表中第一个库的相关共享库，接着搜索列表中的第二个库的相关共享库，依此类推。

版本控制

由于共享库中的代码在运行时将从单独的共享库文件中映射，因此对共享库的修改可能会改变现有可执行文件的行为。在某些情况下，可能导致程序错误操作。

库级别版本

用户可使用命名约定 **libname.***n* 控制其库的版本，其中 *n* 是数字，随着库的新版本而递增。使用新命名方案时，用户在构建共享库时必须使用 **ld** 的 **+h** *internal_name* 选项 为共享库指定内部名称。此内部名称记录在与共享库链接的每个不完整可执行文件或共享库中。

运行时，加载程序将查看记录在不完整可执行文件或共享库中的库列表。对于列表中不是内部名称的库，动态加载程序将查找版本为 **.0** 的库（例如， **libname.0** ）以进行加载。如果未找到该版本，则其将查找记录在列表中的库名称。

显式加载和绑定

上述动态加载程序的任务都是自动执行的，虽然可以通过 **ld** 的相应选项控制它们。动态加载程序还可以编程方式访问。 *shl_load*(3X) 、 *dlclose*(3C) 、 *dlerror*(3C) 、 *dlget*(3C) 、 *dlmodinfo*(3C) 、 *dlopen*(3C) 和 *dlsym*(3C) 下介绍的例行程序提供可移植接口，允许程序员在运行时将共享库显式添加到进程、计算共享库中定义符号的地址和在完成时分离库。

全局符号表

全局符号表机制作为性能增强的选项而设计。启用此机制会创建全局符号表，该表将通过在查找符号时减少扫描全部加载库的需要，加快符号查找的速度。这对于包含大量共享库的应用程序尤其有效。此机制缺省关闭。

全局符号表使用散列表来实现。此机制下，每当加载库时（不论是隐式加载或使用 **dlopen()** 或 **shl_load()** ），此机制就会散列库的输出并将它们放入此表中。卸载库时，此机制将在表中查找库的输出并将其删除。

散列表不包含 **shl_definesym()** 定义的符号条目。因此用户定义的符号必须单独处理。

通过启用此机制， **dld**

可使用更多内存并影响 **dlopen()** 、 **dlclose()** 、 **shl_load()** 和 **shl_unload()** API 调用的性能。

使用全局符号表，动态加载程序可能需要执行大量的的散列操作以定位符号。执行此散列函数可能需要消耗大量时间，尤其在符号名称非常长的时候（C++ 程序）。要加快执行 **dld** 计算，可以从链接程序中卸载散列值。

使用 **+gst** 选项， **+gst** 、 **+gstsize** 和 **+nodynhash** 将控制全局符号表散列机制的行为。有关这些选项的信息，请参阅 *ld*(1) 和 *chatr*(1) 命令。

使用这些选项，用户可以调整 *size* ，以实现性能和内存使用的平衡。要获得最佳性能，请调整某个平均链长度的表的大小。要降低性能以获得最大内存使用，请调整表的大小以使空条目的数量最小。通常，使用质数作为表的大小。

要获得有关散列表性能的统计数据，请设置环境变量 **_HP_DLDOPTS** 以包含 **-symtab_stat** 选项。此选项为每个包含下列信息的库提供一条消息：

- 操作（加载/卸载）
- 库名称
- 输出数
- 表中无存储符号条目的数量
- 非零链的平均长度
- 散列表的计算的性能
- 散列表使用的内存数量

PA-RISC 32 位兼容模式下的动态加载程序的行为

动态加载程序保留一定的行为以支持与较早的 PA-RISC 32 位版本兼容。这些操作适用于以 **ld +compat** 命令和 **shl_load()** 库管理例行程序创建的程序。

动态路径列表

有两种方式指定动态路径列表：

- 通过使用 **ld** 的 **+b** *path_list* 选项在可执行文件中存储目录路径列表。
- 通过使用 **ld** 的选项 **+s** 链接可执行文件，同时允许可执行文件在运行时使用 **SHLIB_PATH** 环境变量定义的路径列表。

路径列表是以冒号 (:) 分隔的一个或多个路径名的列表。动态路径列表仅适用于以 **-l** 或 **ld** 的 **-l:** 选项指定的库。但是通过使用 **chatr** 的 **-l** 选项，也可适用于用完整路径名指定的库（请参阅 *chatr*(1) ）。如果同时使用了 **+s** 和 **+b** ，则它们在命令行上的相对顺序将指示在兼容模式下首先搜索的路径列表。有关详细信息，请参阅 *ld*(1) 的 **+help** 选项或《HP-UX Linker and Libraries User's Guide》。

纵向优先搜索

这是兼容模式下使用的搜索行为，在执行 **shl_load()** 或不完整可执行文件与 **+compat** 相链接时使用。动态加载程序将搜索不完整可执行文件，接着库加载列表中的第一个库。然后搜索此库的第一个相关库，接着搜索相关库的第一个相关库，依此类推。如果没有更多的相关库，则搜索同属库及其相关库，直至最终搜索到程序的库加载列表的第二个库为止，然后搜索该库的第一个相关库，依此类推。

诊断和警告

在兼容模式下，如果希望看到全部信息，请设置环境变量 **_HP_DLDOPTS** 以包含一个或多个选项。支持以下选项：

-warnings 显示附加动态加载程序警告消息。其中一些消息包括：

- 同名但不同类型的符号，如 CODE 和 DATA。有关此警告的详细信息，请参阅 *ld*(1) 中的警告一节。
- 使用 *shl_load*(3X) 描述的某些标志或例行程序。

-fbverbose 请参阅 *fastbind*(1) 。

-nofastbind 请参阅 *fastbind*(1) 。

LD_PRELOAD 环境变量

注释： **LD_PRELOAD** 功能在 seteuid/setegid 程序下禁用，如 **passwd** 。有关详细信息，请参阅 *ld*(1) 。此功能不能用于完全绑定的静态可执行文件。

通过 **LD_PRELOAD** ，用户可以在程序启动时加载附加共享库。 **LD_PRELOAD** 提供动态加载程序可以解释的冒号分隔或空格分隔的共享库列表。动态加载程序 **dld.so** 加载指定共享库，类似于此程序在该程序的其他相关库之前已经在 **LD_PRELOAD** 中与共享库显式链接。

启动时，动态加载程序（如果找到的话）将隐式加载 **LD_PRELOAD** 环境中指定的一个或多个库。使用相同的加载顺序和符号解析顺序，类似于已将库作为创建可执行文件时链接行中的第一个库进行显式链接。例如，给定一个使用下列链接行构建的可执行文件：

$ ld ... lib2.so lib3.so lib4.so

如果 **LD_PRELOAD="/var/tmp/lib1.so"** ，则动态加载程序将使用相同加载顺序和符号解析顺序，类似于已将 **lib1.so** 指定为链接行的第一个库：

$ ld ... /var/tmp/lib1.so lib2.so lib3.so lib4.so

在典型的命令行使用中（使用 **/usr/bin/sh** ），其中 **LD_PRELOAD** 定义如下：

$ LD_PRELOAD=mysl.so *application*

动态加载程序将根据 **$PATH** 搜索应用程序，但根据 **SHLIB_PATH** 和（或） **LD_LIBRARY_PATH**, 和（或）嵌套路径（如果启用）搜索 **mysl.so** 。

注释：因为动态加载程序在运行任何可执行文件（seteuid/setegid 程序除外）时将检查 **LD_PRELOAD** 环境变量，所以如果要导出 **LD_PRELOAD** ，必须在运行可执行文件后取消设置该环境变量，或者在上文列举的命令或脚本中运行可执行文件。

可以使用 **LD_PRELOAD** 环境变量加载共享库，该共享库使用包含了 TLS 的静态线程本地存储模式构建，以避免在动态加载库时出现下列错误：

/usr/lib/hpux[32|64]/dld.so: Can't shl_load() a library containing
Thread Local Storage: /usr/lib/hpux[32|64]/libcps.so.1

可以使用 **+tls=dynamic** 编译程序选项重新编译库，以避免上述错误消息。

加载顺序和符号解析顺序在 PA-RISC 32 位兼容模式下可能会有所不同，这是因为动态加载程序在 PA-RISC 32 模式下使用纵向优先搜索顺序，在标准模式下使用横向优先搜索顺序。有关详细信息，请参阅 *ld*(1) 的 **+help** 选项中的符号搜索和相关库，或者《HP-UX Linker and Libraries User's Guide》。

动态加载程序使用 **LD_PRELOAD** 环境变量，即使在链接行中使用 **+noenvvar** 也是如此。这可确保即使在 **+compat** 链接中也会启用 **LD_PRELOAD** 。除 setuid 和 setgid 程序中外，将始终启用 **LD_PRELOAD** 变量。

注释：如果与混合了共享库和归档库的应用程序一起使用，尤其是当共享库和应用程序都是以 **aC++** 构建，或同时使用 **libc** 时，使用 **LD_PRELOAD** 可导致进行核心转储。

可以指定多个库为 **LD_PRELOAD** 环境变量的一部分。使用空格或冒号分隔 **LD_LIBRARY_PATH** 中的库（未提供多字节支持作为分析 **LD_PRELOAD** 库列表的一部分）。可以使用绝对路径或相对路径指定 **LD_PRELOAD** 库。 **LD_PRELOAD** 库也可以仅由库名组成，在这种情况下，动态加载程序将使用环境变量 **LD_LIBRARY_PATH** 和（或） **SHLIB_PATH** 或嵌套路径列表（如果启用的话）中的目录路径列表来搜索库。

如果动态加载程序无法找到 **LD_PRELOAD** 指定的库，则不发出错误或警告消息。但是，如果它未找到 **LD_PRELOAD** 库的相关库，则动态加载程序将发出同一错误信息，类似于在链接行中指定 **LD_PRELOAD** 。

诊断信息

如果由于任何原因动态加载程序不存在或无法被进程调用，则将错误信息输出到标准错误，并且进程终止，同时返回非零的退出代码。

这些错误可归结为两个基本类别：连接共享库错误和绑定符号错误。前者仅在进程启动时出现，但后者可在进程

执行期间随时出现，除非使用了 **ld** 的 **-B immediate** 选项。在连接到共享库时可能出现的错误包括库不存在、库无法执行、库损坏、高水印过低或库的地址空间不足。绑定符号时可能出现的错误包括符号未找到（未解析的外部引用）或库损坏。

使用动态加载程序的显式加载功能时，这些不是致命错误类型。有关详细信息，请参阅 *shl_load*(3X) 、 *dlclose*(3C) 、 *dlget*(3C) 、 *dlgetname*(3C) 、 *dlmodinfo*(3C) 、 *dlopen*(3C) 和 *dlsym*(3C) 。要参阅错误消息，请使用 **dlerror()** 例行程序。此例行程序将输出动态加载程序记录的最后一条错误消息。

警告

动态加载程序启动的开销非常大，即使使用延时绑定也是如此，并且在启动开销为主的进程中可能导致严重的性能下降（例如简单“hello world”程序）。此外，位置独立的代码通常比普通代码执行速度要慢，因此共享库中的 PIC 对程序的性能可能会造成不利影响。但是在大多数情况下，减少使用磁盘空间和降低可执行文件的内存需求的好处比这些顾虑更为重要。

很少会出现使用共享库和使用归档库时，程序行为不同的情况。这种情况主要发生在依赖编译程序、汇编程序和链接程序的未记录或不支持的功能时。有关详细信息，请参阅 *ld*(1) 的 **+help** 选项或《HP-UX Linker and Libraries User's Guide》。

库开发人员对版本控制负有全责，并且必须严格标志对库接口的不兼容更改。否则，程序在库的后来版本中可能会意外地运行错误。如果库开发人员没有正确处理版本控制，则应用程序用户将对错误无能为力。应用程序开发人员通常可以通过修改源代码，使用新接口来解决问题，然后针对新库重新编译和重新链接。

缺省情况下，动态加载程序不会报告大多数的警告。如果希望查看所有错误消息，请将环境变量 **DLD_VERBOSE_ERR** 设置为 true。

作者

/usr/lib/hpux64/dld.so 和 **/usr/lib/hpux32/dld.so** 共享库由 HP 开发。

另请参阅

系统工具

aCC(1)	调用 HP-UX aC++ 编译程序
as(1)	将汇编代码转换为机器代码
cc(1)	调用 HP-UX C 编译程序
chatr(1)	更改程序的内部属性
f90(1)	调用 HP-UX Fortran 90 编译程序
fastbind(1)	调用 fastbind 工具
ld(1)	调用链接编辑器

其他信息

a.out(4)	汇编程序、编译程序和链接程序输出
dlclose(3C)	卸载先前由 **dlopen()** 加载的共享库
dlerror(3C)	输出 **dld** 记录的最后错误信息
dlget(3C)	返回有关加载模块的信息

dlgetname(3C)	返回包含加载模块的存储名称
dlmodinfo(3C)	返回有关加载模块的信息
dlopen(3C)	加载共享库
dlsym(3C)	获取共享库中符号的地址
shl_load(3X)	加载/卸载共享库

文本和教程

《HP-UX Linker and Libraries Online User Guide》
（请参阅 *ld*(1) 的 **+help** 选项）

《HP-UX Linker and Libraries User's Guide》
（有关订购信息，请参阅 *manuals*(5) ）

名称

dlpi_max_clones - 系统上允许的最大克隆 DLPI 流数量

值

保证安全

256

缺省值

3992

允许值

256 至 **4294967295**

建议值

3992

说明

此变量用于限制系统上可以打开的克隆 DLPI 流的数量。有关 DLPI 的一般信息可从 *dlpi(7)* 联机帮助页中找到。

此可调参数用于防止无权限用户通过打开大量克隆 DLPI 流来恶意地耗用系统资源。

更改此可调参数的人员

权限用户。

更改限制

对此可调参数的更改将在下次重新引导时生效。

应该何时增加此可调参数的值

当 DLPI 应用程序的数量大于缺省值时，应增加此可调参数的值。

增加此值的负面影响

可能的负面影响是可能会降低系统的性能。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装的内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

dlpi_max_clones 由 HP 开发。

另请参阅

kctune(1M)、 gettune(2)、 settune(2)、 dlpi(4)。

名称

dma32_pool_size - 为 32 位 DMA 池保留的内存容量

值

最小值

0

最大值

4GB

缺省值

256MB

指定一个整数值。

说明

此可调参数指定了要为 32 位卡的 DMA 保留的最初 4GB 字节物理地址空间中的内存容量。在引导时将预留该内存，并且不可将其用于其他目的。仅在一些 Itanium 平台上要求如此。在不需要 32 位池的平台上，将忽略该值。

此可调参数的值应基于系统中 32 位卡的数量和卡的类型来设置。

相关参数

无

警告

所有 HP-UX 内核可调参数都是针对于特定版本的。在将来的 HP-UX 版本中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值影响的信息，请参阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅网站 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

dma32_pool_size 由 HP 开发。

另请参阅

无

名称

dnlc_hash_locks - 目录名称查找缓存 (DNLC) 的锁的数量

值

保证安全

512

缺省值

512

允许值

允许的最小值为 **16** 。允许的最大值为 **8192** 。由于该值必须是 2 的幂，并且它必须小于或等于 DNLC 条目 **(ncsize >= 8 * dnlc_hash_locks)** 的八分之一，因此将对其进行进一步限制。指定一个正整数值。

说明

要加快搜索内核中的目录，内存中应驻留一个名为 **Directory Name Lookup Cache (DNLC)** 的目录缓存。在内核中查找文件名的过程中，遇到的所有目录和文件都保存在 DNLC 中以供将来参考。 DLNC 以近期最少使用 **(LRU)** 的方式管理。 **dnlc_hash_locks** 表示将用于管理 DNLC 散列链中的目录条目的锁组数。

更改此可调参数的人员

HP-UX 系统管理员。

更改限制

此可调参数是静态的。对此可调参数的任何更改都在下次系统重新引导后生效。

应该何时增加此可调参数的值

当运行需要进行大量文件名访问的应用程序时，可能需要增大此值来减少锁的争用，并提高文件名的查找性能。

增加此值的负面影响

会消耗更多系统内存来为 DNLC 分配锁。

应该何时降低此可调参数的值

可以减小此值来限制为 DNLC 分配的锁的数量，并减少内存消耗。

降低此值的负面影响

系统重新引导时 DNLC 锁将消耗更少的系统内存。要求文件名查找的操作（例如 **open()** （请参阅 *open*(2) ））可能会遇到性能下降的情况。

应该同时更改的其他可调参数值

ncsize 的值必须等于或大于 **dnlc_hash_locks** 的八倍 **(ncsize >= 8 * dnlc_hash_locks)** 。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在

用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

dnlc_hash_locks 由 HP 开发。

另请参阅

kctune(1M)、 smh(1M)、 gettune(2)、 settune(2)。

名称

dontdump - 定义出现内核混乱时未转储的内核内存页面类别

值

保证安全

0

缺省值

0 （允许内核选择要转储的类别）。

允许值

0 到 **1024** 之间的整数值。

该整数值应当为下面所示类别之外的整数值的组合:

UNUSED	2 : 未使用页
USERPG	4 : 用户页
BCACHE	8 : 缓冲区缓存
KCODE	16 : 内核文本页
USTACK	32 : 进程堆栈
FSDATA	64 : 文件系统元数据
KDDATA	128 : 内核动态数据
KSDATA	256 : 内核静态数据
SUPERPG	512 : 未使用的超级页池

建议值

0 （允许内核选择要转储的类别）。

内核通常选择的值为 **30 = UNUSED + USERPG + KCODE + BCACHE** 。内核选择这些类别是因为它们在调试内核问题时不常使用。

说明

在大型系统中，出现内核混乱时转储系统内存会需要大量、甚至是不可接受的时间，具体取决于系统中安装的物理内存大小。 **dontdump** 和 **alwaysdump** 参数控制的快速转储功能提供了一种限制内核转储到特定信息类型的方法:

. 未使用的物理内存
. 用户进程
. 缓冲区缓存
. 内核代码
. 进程堆栈
. 文件系统元数据

. 内核动态数据
. 内核静态数据
. 未使用的超级页池

crashconf 命令及其关联的配置文件 **/etc/rc.config.d/crashconf** 控制出现内核混乱时内存转储中所包含的内存类别。在个别情况下，在引导进程期间，系统可能会在 *crashconf*(1M) 运行前出现混乱。此时，使用 **alwaysdump** 和 **dontdump** 可调参数可以设置配置。

存储在 **dontdump** 中的位图值指定出现内核混乱时内存转储中将不包含的内存类别。

此参数的缺省值为 **0** 。此时，系统决定是否根据出现的崩溃类型进行一定类别的内存转储。

请注意，特定类型的系统崩溃需要进行完全的崩溃转储。同时，系统运算符在出现转储时会请求完全的崩溃转储。在任一情况下都将执行完全转储，忽略使用 **dontdump** 选择的类别。

更改此可调参数的人员

仅有 HP 现场工程师能更改此可调参数值。

更改限制

对此可调参数的更改将在下次重新引导时生效。要立即生效，请使用 **crashconf** 更改选择的页面。

应该何时启用此可调参数选项

出现系统崩溃时，应该启用该可调参数，以防止对某些类别的页面进行转储。这可以加速转储。

启用此可调参数的负面影响

如果用于分析转储的页面被排除在外，则转储不能帮助查找系统崩溃的原因。

应该何时禁用此可调参数选项

缺省情况下将禁用此可调参数。

禁用此可调参数的负面影响

系统将根据崩溃类型确定不得转储的页面类别。转储可能需要更长的时间。

应该同时更改的其他可调参数

alwaysdump 可调参数不应包含与 **dontdump** 相同的页面类别。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

dontdump 由 HP 开发。

另请参阅

crashconf(1M)、 alwaysdump(5)。

名称

dump_compress_on - 选择出现内核混乱时系统是转储压缩的内存页面还是转储不压缩的内存页面

值

保证安全

0

缺省值

1 （允许内核根据崩溃时的系统状态决定是否转储压缩的内存页面）。

允许值

整型值 **0** 和 **1**。

dump_compress_on 设置使用如下:

0 不压缩内存页面进行转储。

1 如果可能，尽量选择压缩内存页面进行转储，这样的速度会更快。

建议值

1 （允许内核选择转储模式）。

内核选择的值通常为 1。

说明

在大型系统中，出现内核混乱时转储系统内存会需要大量、甚至是不可接受的时间，具体取决于系统中安装的物理内存大小。采用压缩页面的转储方式能使转储过程更小更快，同样在保存至文件系统时花费的时间也更少。

crashconf 命令及相关配置文件 **/etc/rc.config.d/crashconf** 控制转储使用的模式。

该参数的缺省值为 **1** 。此时，系统将根据出现的崩溃类型确定是否转储压缩的内存页面。

请注意，特定类型的系统崩溃需要进行不压缩崩溃转储。另外，系统运算符在出现转储时会请求进行不压缩崩溃转储。在出现上面任一情况时都将执行不压缩内存页面的转储，而不管已在 **dump_compress_on** 中选择的模式如何。

更改此可调参数的人员

仅有 HP 现场工程师能更改此可调参数值。

更改限制

对此可调参数的更改立即生效。

应该何时启用此可调参数选项

在发生系统崩溃时，应启用此可调参数来转储压缩的内存页面。

禁用此可调参数的负面影响

在这种情况下始终采用不压缩转储。在采用大容量内存的计算机上，不压缩的转储使用的时间通常是压缩转储所用时间的三倍。

应该何时禁用此可调参数选项

如果压缩转储出现问题，则应禁用此可调参数。

启用此可调参数的负面影响

系统根据崩溃类型决定转储模式。

应该同时更改的其他可调参数

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

dump_compress_on 由 HP 开发。

另请参阅

crashconf(1M)、 alwaysdump(5)、 dontdump(5)。

名称

dump_concurrent_on - 启用（或禁用）出现内核混乱时系统使用多个转储单元转储内存的选项

值

保证安全

0 禁用

缺省值

1 启用

允许值

0 禁用多个转储单元（同时禁用）

1 启用多个转储单元（同时启用）

建议值

1 允许内核选择是否进行并发转储

说明

在大型系统上，出现内核混乱时转储系统内存会需要大量、甚至是不可接受的时间，具体取决于系统中安装的物理内存的大小。并发转储功能通过将任务分为可并行执行的不同转储单元，从而加速了转储。

dump_concurrent_on 的缺省值为 **1** 。系统根据系统上的可用资源（CPU 和磁盘）和驱动程序的属性决定是否使用多个单元转储。

crashconf 命令还可用来指定是否使用多个转储单元。它将此信息存储在配置文件 **/etc/rc.config.d/crashconf** 中。

更改此可调参数的人员

仅由 HP 现场工程师更改此可调参数的值。

更改限制

对此可调参数的更改立即生效。

应该何时启用此可调参数选项

如果有足够的资源，在系统发生崩溃时，此可调参数应当打开，允许使用多个转储单元进行转储。

禁用此可调参数的负面影响

转储可能需要较长时间。

应该何时禁用此可调参数选项

如果使用多个转储单元时出现问题，则应关闭此可调参数。

启用此可调参数的负面影响

系统根据可用的资源决定是否使用多个转储单元。

应该同时更改的其他可调参数

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

dump_concurrent_on 由 HP 开发。

另请参阅

crashconf(1M)、 alwaysdump(5)、 dontdump(5)、 dump_compress_on(5)。

名称

enable_idds - 启用入侵检测数据源

值

保证安全

0（禁用）

缺省值

0（禁用）

允许值

0（禁用）或 **1**（启用）

建议值

如果安装了 HP-UX HIDS，则为 **1**（启用）

否则为 **0**（禁用）。

说明

注释：从 HP-UX 11i v3 开始， **enable_idds** 可调参数将替换为动态可调参数 *audit_track_paths*(5)。

如果 **enable_idds** 设置为 **1** ，则 HP-UX 主机入侵检测系统 (HP-UX HIDS) 可以启用内核数据集合来进行入侵检测。这也导致内核要跟踪其他的内容，从而使性能稍微降低（同时使用更多内核内存），即使没有使用 HP-UX HIDS 也是如此。

更改此可调参数的人员

使用 HP-UX HIDS 的任何用户。

更改限制

对此可调参数的更改将在下次重新引导时生效。

应该何时启用此可调参数选项

如果安装了 HP-UX HIDS，则此可调参数应该设置为 *on* 。此安装将自动启用 **enable_idds** 。

启用此可调参数的负面影响

跟踪每个进程的当前工作目录（以及根目录）的名称，从而使内存使用和系统性能发生更改。

应该何时禁用此可调参数选项

如果没有正在使用 HP-UX HIDS，则应将 **enable_idds** 设置为 *off* 。

禁用此可调参数的负面影响

当设置为 *off* 时，HP-UX HIDS 无法使用任何使用 **idskerndsp** 的检测模板（有关 **idskerndsp** 的详细信息，请参阅 HP-UX HIDS 文档）。

应该同时更改的其他可调参数

此可调参数与其他可调参数无关。

警告

此可调参数已替换为 **audit_track_paths** 。

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

enable_idds 由 HP 开发。

另请参阅

audit_track_paths(5)、 ids.cf(5)、《HP-UX Host Intrusion Detection System Administrator's Guide》。

名称

environ - 用户环境

说明

进程开始时，名为 **environment** 的字符串数组可以由 *exec*(2) 使用。按照惯例，这些字符串格式为 *name*=*value* 。下列名称可由各种命令使用（以字母顺序排列）：

HOME 用户登录目录的名称，通过口令文件中的 *login*(1) 设置（请参阅 *passwd*(4) ）。

LANG 标识用户要求的本地语言、本地惯例和编码字符集（如果环境变量 **LC_ALL** 、 **LC_COLLATE** 、 **LC_CTYPE** 、 **LC_MESSAGES** 、 **LC_MONETARY** 、 **LC_NUMERIC** 和 **LC_TIME** 未设置或设置为空的话）。

LANG 的格式为：

LANG=*language*[*_territory*][**.***codeset*]

LANG 的有效值为支持的语言环境（请参阅 *lang*(5) ）。通过调用 *setlocale*(3C) ，可在运行时启动本地语言环境支持 (NLS)。下列对 **setlocale()** 的调用将程序的执行绑定到用户的语言要求：

setlocale(LC_ALL,"");

此 **setlocale()** 调用在与 **setlocale()** 相关联的环境变量中初始化程序语言环境。如果任何特定类别的环境变量未设置或设置为空字符串，则 **LANG** 将提供必需的缺省值。

LANG 环境变量最大长度可以为 **SL_NAME_SIZE** 字节（请参阅头文件 **<locale.h>** ）。

LANGOPTS

以下列格式定义模式的语言选项和数据顺序：

LANGOPTS=[*mode*][*_order*]

LANGOPTS 的值作为 ASCII 字符串以英文给定。 *mode* 描述文件模式，其中 **l** (ell) 表示拉丁模式， **n** 表示非拉丁模式。对于除 **l** 和 **n** 以外的值，将采用非拉丁模式。 *order* 说明文件的数据顺序，其中 **k** 表示键盘顺序， **s** 表示屏幕顺序。

LC_ALL 确定所有语言环境类别的值。 **LC_ALL** 的值优先级高于其他任何环境变量 **LC_COLLATE** 、 **LC_CTYPE** 、 **LC_MESSAGES** 、 **LC_MONETARY** 、 **LC_NUMERIC** 、 **LC_TIME** 和 **LANG**。

LC_COLLATE、**LC_CTYPE**、**LC_MESSAGES**、**LC_MONETARY**、**LC_NUMERIC** 和 **LC_TIME**

确定用户的下列要求：语言、地域以及与字符排序、字符分类和转换、输出消息、货币符号和货币价值格式、数值数据表现形式和时间格式分别有关的代码集。如果在环境中未定义 **LC_ALL** 和其中任意一项，则 **LANG** 将提供缺省值。

环境变量 **LC_COLLATE** 、 **LC_CTYPE** 、 **LC_MESSAGES** 、 **LC_MONETARY** 、 **LC_NUMERIC** 和 **LC_TIME** 的语法为：

language[*_territory*][**.***codeset*][@*modifier*]

language 字段遵循语言名称 ISO 639 标准， *territory* 字段遵循地域名称 ISO 3166 标准。有关语言环境名称的列表，请参阅 *lang*(5) 。

通过 @*modifier* 字段，用户可在同一语言定义内，在多个类别值之间进行选择。 HP-UX 当前未向语言环境提供修饰符。

语言环境类别的值由优先级顺序确定；下列满足的第一个条件将确定该值:

1. 如果 **LC_ALL** 环境变量已经定义并且不为空，则使用 **LC_ALL** 的值。
2. 如果 **LC_*** 环境变量（**LC_COLLATE** 、 **LC_CTYPE** 、 **LC_MESSAGES** 、 **LC_MONETARY**、 **LC_NUMERIC** 、 **LC_TIME** ）已经定义并且不为空，则将使用该环境变量的值初始化与环境变量对应的类别。
3. 如果 **LANG** 环境变量已经定义并且不为空，则使用 **LANG** 环境变量的值。
4. 如果 **LANG** 环境变量未设置或者设置为空字符串，则使用 POSIX/C 缺省语言环境（请参阅 *lang*(5) ）。

LC_COLLATE
用于确定字符排序的语言环境类别。该项将确定在各种实用程序、 *strcoll*(3C) 和 *strxfrm*(3C) 中正则表达式和排序的排序信息，包括等效类和多字符排序元素（请参阅 *string*(3C) ）。

LC_CTYPE 确定用于字符分类的语言环境类别（例如字母、数字、大写）。请参阅 *isalpha*(3C) 、 *isdigit*(3C) 和 *isupper*(3C) ，以及 *ctype*(3C) 中的字符转换。请参阅 *toupper*(3C) 、 *tolower*(3C) 和 *conv*(3C) 中解释为单字节或多字节字符的文本解释。

LC_MESSAGES
用于确定用于处理肯定和否定响应的语言环境类别，以及输出诊断消息和信息性消息的语言和文化惯例。它在确认打开消息清单时还会影响 *catopen*(3C) 的行为。

LC_MONETARY
确定货币相关数字格式设置信息的语言环境类别。

LC_NUMERIC
用于确定以下内容的语言环境类别：各种实用程序中的数字格式设置信息（例如千位分隔符和基数字符）， *printf*(3S) 和 *scanf*(3S) 中的格式设置的 I/O 操作，以及 *strtod*(3C) 中的字符串转换函数。

LC_TIME 确定日期和时间格式设置信息的语言环境类别。它将影响 *strftime*(3C) 中时间函数的行为。

MANPATH 包含以冒号分隔的目录前缀列表， *man*(1) 将对其进行搜索以查找手册条目。在登录时， **/etc/profile** （或 **/etc/csh.login** ）将设置 **MANPATH=/usr/share/man:/usr/contrib/man:usr/local/man** 。如果存在文件 **/etc/MANPATH** ，则采用此文件中的缺省设置。

MANPATH 使用与 **PATH** 环境变量相同的语法，此外还可识别在 **NLSPATH** 环境变量中使用的说明符 **%L** 、 **%l** 、 **%t** 和 **%c** 。有关这些说明符的说明，请参阅下面的 **NLSPATH** 。它提供了指定特定语言环境的手册条目的路径的方法。

假定 **MANPATH** 中给定的每个前缀都包含 **man*** 、 **man*.Z** 、 **cat*** 和 **cat*.Z** 格式的子目录（请参阅 *man*(1) 、 *catman*(1M) 和 *fixman*(1M) ）。

NLSPATH 包含在尝试查找消息清单时， *catopen*(3C) 使用的伪路径名序列。每个伪路径名包含一个名称模板，该模板由一个可选路径前缀、一个或多个替换字段描述符、一个文件名和一个可选文件名后缀组成。例如：

NLSPATH="/system/nlslib/%N.cat"

定义 *catopen*(3C) 应查找目录 **/system/nlslib** 中的所有消息清单，其中清单名称应根据 *name* 参数构建，该参数将传递给前缀为 **.cat** 的 *catopen* (3C) (%N)。

字段描述符由 **%** 其后跟单个字符组成。字段描述符及其替换值包括：

%N	*name* 参数的值传递给 *catopen*(3C) 。
%L	**LC_MESSAGES** 的值。
%l	**LC_MESSAGES** 中的 *language* 元素。
%t	**LC_MESSAGES** 中的 *territory* 元素。
%c	**LC_MESSAGES** 中的 *codeset* 元素。
%%	使用一个单独的 **%** 进行替换。

例如，给定：

NLSPATH="/system/nlslib/%L/%N.cat"

catopen(3C) 尝试将文件 **/system/nlslib/$LC_MESSAGES/***name***.cat** 打开为消息清单。

如果未定义指定的值，则替换空字符串。替换 **%t** 和 **%c** 时不包含分隔符。请注意，未向 **%L** 提供缺省值。如果未设置 **LC_MESSAGES** 并且 **NLSPATH** 具有前例中的值，则 *catopen*(3C) 将尝试将文件 **/system/nlslib/***name***.cat** 打开为消息清单。

NLSPATH 中定义的路径名以冒号 **(:)** 分隔。前导的冒号或两个相邻的冒号 **(::)** 等效于指定 **%N** 。例如，给定：

NLSPATH=":%N.cat:/nlslib/%L/%N.cat"

oflag 参数设置为 **NL_CAT_LOCALE** 的 *catopen*(3C) 将尝试按指示的顺序打开下列文件： **./***name*, **./***name***.cat** 和 **/nlslib/$LC_MESSAGES/***name***.cat** 。第一个成功打开的文件将作为消息清单。

每次在用户定义的伪路径名中无法打开消息清单时，系统定义的缺省伪路径名将有效地追加到 **NLSPATH** 之后，并由 *catopen*(3C) 进行使用。系统级缺省路径为：

/usr/lib/nls/msg/%L/%N.cat:/usr/lib/nls/%l/%t/%c/%N.cat

如果以所有者超级用户权限从 **setuid** 或 **setgid** 应用程序中调用 *catopen*(3C) ，则不使用环境变量 **NLSPATH** 。而改用系统文件 **/etc/default/nlspath** 查找消息清单。有关详细信息，请参阅 *nlspath*(4)。

PAGER **PAGER** 表示通过其以管道传输某些命令的输出的分页程序。其值必须是指定所需分页程序的完整命令行的字符串。以下为两个示例:

PAGER="more -cs"
PAGER="pg -c"

PAGER 可影响多个命令，包括 *man*(1) 和交互式邮件程序。一些受影响的程序在发生冲突时，提供了选择寻呼的替换方法。有关详细信息，请参阅单独的手册条目。

PATH **PATH** 表示查找不完整路径名已知的文件时， *sh*(1) 、 *time*(1) 、 *nice*(1) 、 *nohup*(1) 和其他项搜索的目录前缀的序列。前缀以冒号分隔 (**:**) 。 *login*(1) 命令可设置 **PATH/usr/bin** 。

TERM **TERM** 标识准备输出的终端类型。此信息由命令（例如 *vi*(1) 和 *mm*(1) ）使用，这些命令可使用该终端的特殊功能。

TZ **TZ** 用于设置时区信息。 **TZ** 可采用如下格式设置:

[**:**]**STD***offset*[**DST**[*offset*][**,***rule*]]

其中:

STD 和 **DST** 使用三个或更多字节指定标准时区 (**STD**) 和夏令时时区 (**DST**) 。 **STD** 是必需的。如果未指定 **DST** ，则夏令时在此语言环境下不适用。这些字段中的每个字段都可能以带引号或不带引号的格式出现。

在带引号的格式中，第一个字符应小于 (**<**) 字符，而最后一个字符应大于 (**>**) 字符。这些带引号字符之间的所有字符应该是来自当前语言环境中的可移植字符集的字母数字字符、加号 (**+**) 字符或减号 (**-**) 字符。在这种情况下， **STD** 和 **DST** 字段不要包含带引号的字符。

在不带引号的格式中，允许使用除数字、逗号 (**,**)、减号 (**-**)、加号 (**+**) 或 ASCII NUL 之外的任何字符。

offset *offset* 是必须加到本地时间上的值，以达到国际标准时间 (UTC)。 *Offset* 格式如下:

hh[**:***mm*[**:***ss*]]

小时 (*hh*) 是从 0 到 23 的任意值。可选的分钟 (*mm*) 和秒 (*ss*) 字段的值从 0 到 59。小时字段是必需的。如果 *offset* 前面是 **-** ，则时区在本初子午线以东。 **+** 在 *offset* 之前表示时区在本初子午线以西。缺省情况为本初子午线以西。

rule *rule* 指定何时更改为或退出夏令时时间。 *rule* 具有以下形式:

*date***/***time***,***date***/***time*

其中第一个 *date***/***time* 指定将标准时间更改为夏令时的时间，第二个 *date***/***time* 指定改回来的时间。 *time* 字段以当前本地时间表示。

date 格式应当为以下之一:

J*n* 儒略日 *n* （1 到 365）。不计算闰日。不能引用 2 月 29 日。

n 基于零的儒略日（0 到 365）。计算闰日。可以引用 2 月 29 日。

*M***m.n.d** 一年中的 *m* 月（1 到 12）第 *n* 个星期（1 到 5）的第 *d* 天（0 到 6）。第五个星期指的是 *m* 月的最后 *d* 天。第一个星期是每个月头一天所在的星期。第 0 天是星期天。

time 除了不可以使用前导符号 (**-** 或 **+**) 之外，*Time* 的格式与 *offset* 相同。缺省情况下，如果未指定 *time* ，其值为 02:00:00。

如果还提供了 DST 字段，则同时必须指定 STD 字段和 STD 的 offset 字段，系统将为未指定的其他字段提供缺省值。这些缺省值来自文件 **/usr/lib/tztab** （请参阅 *tztab*(4)），并且通常反映了夏令时起始时间的各种历史数据。

其他名称可以通过 *export* 命令和 *sh*(1) 中的 *name***=***value* 参数，或通过 *exec*(2) 放置在环境中。不要添加与下列 Shell 变量（通常由 **.profile** 文件导出）冲突的名称： **MAIL** 、 **PS1** 、 **PS2** 和 **IFS** 。

在 C 中使用全局变量可以访问进程的环境。

```
#include <unistd.h>

extern char **environ;
```

指向包含环境的字符串指针数组。该数组以空指针结尾。

注释

国际标准时间 (UTC) 与格林威治标准时间 (GMT) 等效。

外部语言环境影响

国际代码集支持

LANG 、 **LC_ALL** 、 **LC_COLLATE** 、 **LC_CTYPE** 、 **LC_MESSAGES** 、 **LC_MONETARY** 、 **LC_NUMERIC** 、 **LC_TIME** 和 **NLSPATH** 环境变量为国际化应用程序提供支持。标准实用程序按此处和实用程序的各环境变量小节中的说明使用这些环境变量。

如果这些变量指定的语言环境类别不是基于相同的基础代码集，则其结果是未确定的，并且正则表达式 API（例如 **regcomp** 、 **glob** 和 **fnmatch** ）可能会受影响。

警告

有些 HP-UX 命令和库例行程序不使用 **LANG** 、 **LC_COLLATE** 、 **LC_CTYPE** 、 **LC_MONETARY** 、 **LC_NUMERIC** 、 **LC_TIME** 或 **LANGOPTS** 环境变量。有些命令不使用消息清单，因此 **NLSPATH** 不会影响它们的行为。有关实现特定命令和库例行程序的详细信息，请参阅“外部语言环境影响”一节。

作者

environ 由 AT&T 和 HP 开发。

另请参阅

env(1)、login(1)、sh(1)、exec(2)、 catopen(3C)、ctime(3C)、getenv(3C)、setlocale(3C)、nlspath(4)、 profile(4)、tztab(4)、lang(5)、term(5)。

符合的标准

environ：AES、SVID3、XPG2、XPG3、XPG4、FIPS 151-2、POSIX.1

名称

eqmem_limit - 确定等效映射内存的最大大小（以 MB 为单位），等效映射内存可以在引导后分配

值

缺省值

系统决定

允许值

0 - 419304

使用明确的值时，这个值将解释为以 MB 为单位的数值。

建议值

系统决定的缺省值。

说明

等效映射内存是具有相同物理地址和虚拟地址的内存。在 PA-RISC 系统上，某些内核结构需要此类型的内存。大部分内存在早期引导时分配，某些内存在添加到系统时分配；等效映射内存也可能用于某些设备驱动程序和一小部分内核侵入应用程序。

引导时，HP-UX 内核选择某些内存作为潜在的等效可映射内存。只能为这些页指定等效映射。等效内存不需要这些页时，这些页可以用于并将用于其他用途，所以通常需要将大量页指定为等效可映射内存。如果将 **eqmem_limit** 设置为缺省值，则内核将计算适当的限制范围。此限制范围将为保守的限制范围，目的是确保其足以支持最大内存量的联机添加 (OLA)。

如果将可调参数设置为非缺省值，则系统将使用该限制范围。

请注意，此可调参数只提供上限。假定为动态内存添加，则将限制范围设置为大于当前现有的内存是合法且正常的。而且，如果内存为不可弹出的内存，并且其物理地址对应于动态内核内存的合法虚拟地址，则此内存仅能用于等效映射内核内存。因此，等效可映射内存的实际数量可能少于 **eqmem_limit** 或系统上的总内存数量。

更改此可调参数的人员

限制使用内存联机添加的 ccNUMA（Cache-Coherent Non-Uniform Memory Access，高速缓存相关的非一致性内存访问）系统。

充分利用内存联机添加，且将重要的等效映射内存用于其他用途的系统。

更改限制

此可调参数仅存在于 PA-RISC 系统。

对此可调参数的更改将在下次重新引导时生效。

何时应更改此可调参数的值

为了在 ccNUMA 系统上获得最好的性能，应当从交叉内存上分配某些共享数据结构。如果所有可用的交叉内存指定为等效可映射内存，且单元本地内存可用，则这些结构将在单元本地内存外分配。

如果系统具有混合的交叉内存和单元本地内存，且不用于执行联机添加内存，或将要添加的总内存远远少于支持

的最大内存，则 **eqmem_limit** 可调参数可用于限制将指定为等效可映射内存的总内存，因此导致从交叉内存分配更多此类结构。

可能还需要重新配置具有较多交叉内存和较少的单元本地内存的系统。

此可调参数值的保守下限为预期联机添加的总内存的 2%，或若干兆字节，取其中的较大值。不太保守的限制值为总内存的 1%（初始值和 OLA 内存，如果不出现 OLA，则为 0，并且已知没有需要等效内存的驱动程序或应用程序）。

对于既不支持联机内存添加，也不支持单元本地内存的系统，更改此可调参数没有特别的作用。对于只有少数位置（单元）的小 ccNUMA 系统，更改此可调参数通常也没有特别的作用。

缺省值极其保守，应当支持比实际可能的内存更多的联机内存添加，并为少量可能需要内存的应用程序和驱动程序保留大量等效可映射内存，但是必须要有足够实际等效可映射（非可弹出、非单元本地，并且在内核动态虚拟地址的合法范围内具有物理地址）的物理内存。但是，指明为等效可映射的内存可能用于其他用途，而且使用模式可能导致在 OLA 需要时内存不可用。如果遇到此类问题，将可调参数值设置为高于计算的缺省值可能有用（仅应当在 HP 客户支持工程师的建议下执行此操作）。

更改此值的负面影响

如果指定为等效可映射内存的总数量太少，分配可能会失败。这可能导致联机添加内存失败，或导致驱动程序需要等效内存的设备失败。

如果指定为等效可映射内存的总数量太高，则访问 ccNUMA 系统上的共享结构时可能会感觉到性能有些降低。

不支持高于 104856（1024 * 1024 MB，即 1 TB）的值，所以除非客户支持工程师建议这样设置，否则应当避免设置为这些值。

应该同时更改的其他可调参数值

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

eqmem_limit 由 HP 开发。

名称

eqmemsize - 确定等效映射保留池的最小大小（以页为单位）（已过时）

说明

此可调参数已过时并被删除。

如果需要控制引导后内核可用的等效映射内存的总量，请使用新的可调参数 **eqmem_limit** （请参阅 *eqmem_limit*(5)）。

请注意，一般而言，如果系统通过设置 **eqmemsize** 就可以满足要求，则无需设置 **eqmem_limit**。

等效映射内存是给定相同物理和虚拟地址的内存。在 PA-RISC 系统上，是支持联机添加内存所必需的，并可能对一些应用程序和一些 I/O 设备非常有用。

HP-UX 11i v2 保留（少量）等效映射页面，它们不能用作其他用途。另外，也可能会等效映射物理地址低于最大内核虚拟地址的任何页面，但前提是系统发现这些页面的虚拟地址和物理地址恰巧同时可用；而除引导后的瞬间外，这种情况几乎不会发生。 **eqmemsize** 可调函数用于调整保留量的大小。它的保留量始终非常小，除在已知将要使用一定数量的内存的系统上例外，在种情况下，保留池大小可以使用 **eqmemsize** 可调参数来增加。

HP-UX 11i v2 之后的版本完全重写了等效内存分配器。当前版本的等效内存分配器用于确定引导时将要等效映射的页面。它可以使相应的虚拟地址不被用作其他用途，从而保证在物理页面可用时，可以对其等效地进行映射。它允许将这些页面用作其他的用途，但仍可以可靠地重新用于等效映射。因此它不需要保留。 **eqmem_limit** 可调参数对将要等效映射的内存的总量设置了上限。

此类页面可视为与其他页面基本相同，但不是完全相同。其差别仅对连贯缓冲非统一内存寻址 (Cache-Coherent Non-Uniform Memory Access, ccNUMA) 系统存在影响，其中在某些情况下这些差别会造成性能的下降。在这种系统上， **eqmem_limit** 可调参数用来减少将被指定用于等效映射的内存总量，使之下降到实际需要的最大量（通常情况下，内核对可能需要的总量的估计非常保守）。有关详细信息，请参阅 *eqmem_limit*(5)。

作者

eqmemsize 由 HP 开发。

另请参阅

eqmem_limit(5)。

名称

EVM - 事件管理

说明

事件和事件管理简介

事件管理系统的用途是为任何系统组件或应用程序提供一种方法，用于指示发生了其他某个实体可能关注的事情。这种指示称为事件，发布事件的组件称为事件生成器或事件发布器。关注指示的实体称为事件订阅者。

如果系统组件有受关注的事情要报告，它将通过事件通道来提供该信息。术语“事件通道”表示任何可用于发布或检索事件信息的工具，可能是下列任一项：

- 简单日志文件
- 事件管理系统
- 为获取状态信息快照而运行的程序

事件管理系统是一个活动的事件通道，因此，它可以提供用于分发、存储和检索事件信息的服务。

HP-UX 系统日志程序 syslog 是一个事件管理系统示例。它可提供简单事件分配工具，供其他组件使用，其守护程序可以主动地管理收到的事件信息。相反， **cron** 守护程序的日志文件 **/var/adm/cron/log** 是一个被动事件通道。 **cron** 守护程序仅在它的文件末尾写入新的事件信息，并且在写入信息时，不会执行特殊的操作来通知关注的实体。

一般而言，事件发布器并不知道关注事件信息的任何实体。它只是使用可用的事件通道来发布事件。事件通道负责确定如何提供事件，以及向谁提供。事件订阅者负责向事件通道确定事件关注点。订阅者可以是用户级进程、内核子系统或（通过某个实用程序订阅的）用户。

关于 **EVM**

事件管理器 (EVM) 是一个综合性的事件管理系统，它除了可以提供传统的事件处理工具外，还可以统一来自多个通道的事件，以提供系统级信息源。有关将 EVM 用作系统管理辅助手段的信息，请参阅《EVM System Administration Guide》。

EVM 事件

EVM 事件是可以在程序之间传递，并可存储在文件中的信息包。事件程序包的基础格式为二进制，但可以使用提供的命令和编程接口来提取和显示事件中包含的信息。术语“原始事件”用于表示二进制状态的事件，为显示而转换为文本格式的事件被称为已设置格式的事件。

EVM 事件可能包含一组标准事件数据项中的任一项或所有项，包括（但不限于）事件名称、时间戳、优先级值和某些消息文本。事件还可能含有任意数目的命名变量数据项，其中每项都可以保存有关事件的更多信息。EVM 事件可能含有来自其他通道（例如二进制错误日志程序）的事件，这些事件保存在变量数据项中。

EvmEvent(5) 联机帮助页中提供了 EVM 事件的全部详细信息。

EVM 守护程序

将系统初始化至级别 2 后，将自动启动 EVM 守护程序 **evmd** 。守护程序为本地系统上运行的系统和应用程序客户端提供发布服务和通知服务。有关详细信息，请参考 *evmd*(1M) 和 *evmdaemon.conf*(4) 联机帮助页。

EVM 日志程序

EVM 日志程序 **evmlogger** 是由守护程序自动启动的事件订阅者。日志程序将读取其配置文件，以建立要记录的事件集，再订阅这些事件，然后在事件到达时将它们存储在管理的日志文件中。缺省情况下，日志程序还会在系统控制台上显示高优先级事件，并通过邮件将有关这些事件的信息发送到超级用户。可配置日志程序，使其管理任意数目的日志文件（每个日志文件都包含其自身选择的事件），以及在收到选定的事件时执行用户提供的命令。

有关详细信息，请参考 *evmlogger*(1M) 和 *evmlogger.conf*(4) 联机帮助页。

EVM 通道管理器

EVM 通道管理器 **evmchmgr** 由守护程序自动启动，它负责管理基于时间的事件通道功能。通道管理器将读取 EVM 通道配置文件，并针对任何配置的被动通道，定期运行事件监视命令。该程序还负责每日运行日志文件清理命令。

evmchmgr(1M) 和 *evmchannel.conf*(4) 联机帮助页中介绍了通道管理器和通道配置文件。

命令行实用程序

EVM 的系统管理工具包括一组命令行实用程序，可用于从命令行或者在 shell 脚本中发布事件、监视事件活动、检索日志文件中存储的事件，以及以不同的方式排序和查看事件。设计这些实用程序旨在与 shell 管道线一起使用。有关详细信息，请参考 *evmpost(1)* 、 *evmwatch(1)* 、 *evmget(1)* 、 *evmsort(1)* 和 *evmshow(1)* 联机帮助页。

过滤事件

由于在一天内，系统可能会生成许多事件，因此，通常需要将查看的内容限制到您所关注的特定事件集。例如，您可能需要查看某个特定子系统发布的事件，或者需要查看具有某个高优先级值的所有事件。可以使用事件过滤器选择 EVM 事件，该过滤器是一个字符串，用于说明使用预定义过滤语法进行选择。可以根据多个不同的条件（包括事件名称、时间戳、优先级和发布系统的名称），使用过滤器选择事件。

可通过在某些 EVM 命令行实用程序上指定 **-f** 选项来使用事件过滤器。 EVM 日志程序在其配置文件中使用事件过滤器，以便发生特定事件时，可以选择要执行的操作。可以将经常使用的事件过滤器存储在过滤器文件中，以便参考。

有关事件过滤语法以及使用过滤器文件的详细信息，请参考 *EvmFilter*(5) 和 *evmfilterfile*(4) 联机帮助页。

事件模板文件

事件模板文件用于控制可在给定系统上发布的事件集，以及为给定的事件中包含的大量信息提供中心源。例如，每次发布某个给定的事件后，该事件的优先级和消息文本都很有可能相同，与将这些信息保存在发布程序或内核中相比，集中这些信息将更易于查看和维护。

事件模板文件是一个文本文件，用于保存一个或多个指定事件的模板信息。只有在安装模板文件后，才可以发布该模板文件说明的事件，并且 EVM 守护程序每次启动或重新加载其配置时，才可以读取这些事件。发布一个事件后，守护程序会将模板中保存的信息添加到发布的事件，然后将事件分配给订阅者。

有关模板文件的用途和语法的详细信息，请参阅 *evmtemplate*(4) 联机帮助页。

事件授权

由于在某些环境中，对监视或发布特定事件的功能不设限制会降低安全性，因此，EVM 为特定的已授权用户提供了一种方法，用于对发布和访问选定事件的功能进行限制。有关详细信息，请参考 *evm.auth*(4) 联机帮助页。

EVM 编程接口

EVM 应用程序编程接口 (API) 库 **libevm.so** 提供了应用程序创建、发布和订阅事件、在标准文件描述符中读取和写入事件、以及处理事件内容所需的全部功能。有关使用 EVM 进行编程的完整介绍，请参考《EVM Programmer's Guide》和“另请参阅”小节中列出的例行程序的联机帮助页。

EVM 支持通过伪设备驱动程序 **/dev/kevm**

在内核空间中进行事件发布和订阅。

另请参阅

命令

evmget(1)、 evminfo(1)、 evmpost(1)、 evmshow(1)、 evmsort(1)、 evmwatch(1)、 evmchmgr(1M)、 evmd(1M)、 evmlogger(1M)、 evmreload(1M)、 evmstart(1M)、 evmstop(1M)。

例行程序

EvmConnControl(3)、 EvmConnCreate(3)、 EvmConnSubscribe(3)、 EvmConnWait(3)、 EvmEventCreate(3)、 EvmEventDump(3)、 EvmEventFormat(3)、 EvmEventPost(3)、 EvmEventRead(3)、 EvmEventValidate(3)、 EvmFilterCreate(3)、 EvmItemSet(3)、 EvmSrvStart(3)、 EvmStatusTextGet(3)、 EvmVarSet(3)

文件

evm.auth(4)、 evmchannel.conf(4)、 evmdaemon.conf(4)、 evmfilterfile(4)、 evmlogger.conf(4)、 evmtemplate(4)。

杂项

sys_attrs_kevm(5)。

事件回调

EvmCallback(5)。

事件连接

EvmConnection(5)。

EVM 事件

EvmEvent(5)。

事件过滤器

EvmFilter(5)。

《EVM Programmer's Guide》。

《EVM System Admnistration Guide》。

名称

EvmCallback() - 事件管理 (EVM) 回调函数

概要

void EventCB(
EvmConnection_t *connection***,**
EvmCallbackArg_t *callbackArg***,**
**EvmCallbackData_t ****callbackData*
);

说明

程序可以使用提供的 EVM 回调函数来处理 EVM 连接上收到的消息。大多数消息是传入的事件，或者是向 EVM 守护程序发出请求后得到的响应，但是，也可能出于其他原因而获得回调。

使用回调的响应模式创建 EVM 连接时，在回调参数中指定回调函数的名称 (*EventCB*) 。

回调函数由 **EvmConnDispatch()** 调用，只要在连接上检测到未决的活动，就必须调用该函数。

EVM 回调函数必须符合显示的原型。

回调函数的参数

- 连接参数包含与此回调有关的连接环境。此值是在创建连接后由 **EvmConnCreate()** 返回的。如果有一个回调函数在为多个连接提供服务，则可以测试此值，以确定特定的调用所处理的连接。
- *callbackArg* 参数包含在创建连接时，指定为 **EvmConnCreate()** 的 *callbackArg* 参数的值。例如，可以使用此参数为连接保存一个指向您自己的环境数据的指针。
- *callbackData* 参数指向一个包含回调详细信息的结构。该结构包括一个回调原因代码、一个事件以及一个联合，该联合包含特定于每个可能的原因的数据。如果事件成员不适用于原因，则可能包含 NULL。可以获取这些原因：

EvmREASON_EVENTS_MISSED

此回调原因指示守护程序无法传递一个或多个事件，这可能是因为客户端进程尚未完成前面的事件处理，从而导致连接缓冲区溢出。

如果必要，可以使用 **EvmConnControl()** 增加接收缓冲区的大小。请参阅 *EvmConnControl*(3) 参考页。

EvmREASON_EVENT_DELIVERED

此回调原因表示从守护程序收到了 EVM 事件。 **callbackData** 结构中将提供该事件。完成该事件的处理后，必须使用 **EvmEventDestroy()** 将其释放。请参阅 *EvmEventDestroy*(3) 参考页。

EvmREASON_POST_COMPLETE

创建连接后，如果 *responseMode* 指定为 **EvmRESPONSE_CALLBACK** ，此回调原因通常使用来自守护程序的响应指示已完成前面的 **EvmEventPost()** 调用。 **callbackData** 结构中将提供发布状态。

EvmREASON_REGISTRATION_DELIVERED

此回调原因指示已传递请求的模板，以响应 **EvmConnRegistrationGet()** 请求。完成该模板的处理后，必须使用 **EvmEventDestroy()** 将其释放。

EvmREASON_SUBSCRIBE_COMPLETE

此回调原因表示之前的 **EvmEventSubscribe()** 调用已完成，并且守护程序通常会对此作出响应。

EvmREASON_TEMPLATE_INFO_DELIVERED

此回调原因指示已传递事件模板信息，以响应 **EvmConnTemplateScan()** 请求。

callbackData 结构中提供了该模板。完成该模板的处理后，必须使用 **EvmEventDestroy()** 将其释放。

文件

/usr/include/evm/evm.h

包含回调函数的事件声明、结构和原型的头文件

另请参阅

例行程序

EvmConnCheck(3)、 EvmConnCreate(3)、 EvmConnSubscribe(3)、 EvmEventPost(3)。

事件管理

EVM(5)。

事件连接

EvmConnection(5)。

EVM 事件

EvmEvent(5)。

《EVM Programmer’s Guide》，

《EVM System Admnistration Guide》。

名称

EvmConnection - 连接到 EVM（事件管理）守护程序

说明

EVM 连接是在 EVM 守护程序中传递数据时所要通过的环境。连接具有下列属性:

- 连接类型
- 响应模式
- 传输类型
- 回调和关联的回调参数
- 连接环境

下列各节将介绍这些属性。

EVM 客户端是通过 EVM 守护程序处理事件的任何程序。 EVM 支持三种不同类型的客户端：发布客户端、订阅（监听）客户端和服务客户端。无论是哪种类型，所有客户端都以相同的方式与 EVM 守护程序连接。

连接类型

建立的连接类型确定客户端类型。创建连接后，需要使用某些关联的常量。

发布连接 **(EvmCONNECTION_POST)**

客户端使用此连接将事件发布到守护程序以进行分配。

监听连接 **(EvmCONNECTION_LISTEN)**

客户端使用此连接监听守护程序分配的事件。

服务连接 **(EvmCONNECTION_SERVICE)**

客户端使用此连接请求守护程序提供服务，例如从日志中检索事件。

一个客户端可以使用所有三种连接类型，但是必须单独建立每种连接。

响应模式

与连接关联的响应模式确定特定的 API 函数在处理守护程序对请求消息做出响应时所采用的方式。有关每种模式的全部详细信息，请参阅 *EvmConnCreate*(3) 参考页。这些模式包括:

忽略 **(EvmRESPONSE_IGNORE)**

将请求发送到守护程序后， API 函数将立即返回，同时，调用方不会收到守护程序的响应。

等待 **(EvmRESPONSE_WAIT)**

API 函数将请求发送到守护程序，然后一直等待，直到收到响应并随后将响应返回给调用方。返回的状态代码将反映该响应。

回调 **(EvmRESPONSE_CALLBACK)**

将请求发送到守护程序后， API 函数将立即返回，同时，调用方必须针对响应监视连接。收到响应后，将调用连接的回调函数来处理该响应。

传输类型

传输类型指定要与守护程序建立的连接类型。下面是可以进行的唯一有效连接:

本地连接 **(EvmTRANSPORT_DOMAIN_SOCKET)**

通过域套接字，与本地主机上运行的守护程序建立连接。

回调

该属性指定处理连接上的活动所产生的任何传入响应时需要使用的函数。 *EvmCallback*(5) 参考页中更详细地介绍了回调。仅当响应模式为 **EvmRESPONSE_CALLBACK** 时，该属性才有效。

连接环境

该属性是创建连接后，返回的连接的句柄。必须将此句柄传递给需要访问连接的其他所有调用的函数。

连接监视

一旦建立了连接，API 函数将使用守护程序处理所有通信活动。但是，必须确保在发生活动时，这些函数有机会执行其自身的任务。EVM 提供多种方法来达到此目的。选择的方法取决于程序的模型。

- 如果程序是 I/O 驱动的，则等待一个或多个文件描述符上出现 I/O，出现 I/O 后将处理活动，然后返回以等待出现其他 I/O，有可能需要将大多数时间花费在 **select()** 调用上。在这种情况下，应使用 **EvmConnFdGet()** 来建立 EVM 连接所使用的文件描述符，然后将该文件描述符包含在 **select()** 读取掩码中。检测连接上的活动时，请调用 **EvmConnDispatch()** 来处理活动。
- 如果程序完全由单个 EVM 连接上的活动驱动，则可以让 API 来完全处理 I/O，方法是使用 **EvmConnWait()** 来等待连接上出现活动。函数返回后，请使用 **EvmConnDispatch()** 发送 I/O，然后返回到 **EvmConnWait()** 。
- 如果程序是通过其他某种方式驱动的，并且您需要在某些点（例如某个控制回路的终点）上处理 EVM 活动，则可以调用 **EvmConnCheck()** 来检查是否存在任何未完成的活动。如果该函数指示需要执行某项操作，则可以调用 **EvmConnDispatch()** ；否则，可以立即继续正常处理。

损坏一个连接

完成连接的处理后，使用 **EvmConnDestroy()** 断开与守护程序的连接。每次调用连接函数时，请检查返回状态以确定是否发生任何失败，如果发生失败，请损坏连接，这一点至关重要。在损坏连接之前，与该连接关联的文件描述符将保持为打开状态，即使在连接上检测到 I/O 错误，也是如此。

另请参阅

函数

select(2)。

例行程序

EvmConnControl(3)、EvmConnCreate(3)、EvmConnFdGet(3)、EvmConnCheck(3)、EvmConnWait(3)、EvmConnDispatch(3)、EvmConnDestroy(3)。

事件管理

EVM(5)。

事件回调

EvmCallback(5)。

EVM 事件

EvmEvent(5)。

名称

EvmEvent - EVM 事件的结构

说明

EVM 事件是独立的数据结构，可以使用 EVM API 函数进行操作和访问。应用程序代码可用于:

- 创建、复制和破坏事件。
- 设置和检索事件中包含的标准数据项的值。
- 向事件添加变量数据项，并设置和检索它们的值。
- 将事件发布到 EVM 守护程序以便分发给用户。
- 从打开文件描述符读取事件。
- 向打开文件描述符写入事件。

提供了命令行实用程序，允许用户访问这些功能。

事件的内容

事件结构包含两种类型的数据项:

1. 预定义名称的标准数据项
2. 变量数据项，其名称和类型在将该数据项添加到事件时定义

   创建事件时，可以根据需要包含任意数量的数据项。当发布事件时，API 函数自动添加适用于当前环境的标准项，如主机名和时间戳。

标准数据项

事件中通常需要标准数据项，并且这些数据项可以由 EVM 识别并执行。下表是事件中可以包含的标准数据项的列表。标识符是用于发布、显示或设置项目格式的关键字。

数据项/Id	说明
Event Name	
NAME	命名事件。如果要发布事件，必须与守护程序的模板数据库中的某一名称匹配。
Time Posted	
TIMESTAMP	首次生成此事件时的 UNIX 系统时间。
Repeat Count	

REPEAT_COUNT 已经合并到单个存储事件中的相同事件的实例数量。 Time Posted 和 Last Timestamp 项指示发布事件的第一个和最后一个实例的时间。

Last Timestamp
LAST_TIMESTAMP 如果存在 Repeat Count 并且其不为零，则生成上次出现此事件时的系统时间。

Event Identifier
EVENT_ID 标识事件。请参阅下表的说明。

Kernel Only
KERNEL_ONLY 如果存在，并且在内核中为事件生成了 **EvmTRUE** ，则该事件不会从内核传送到用户空间。

Process ID
PID 发布事件的进程的 PID。

Parent Process ID
PPID 发布事件的进程的父进程 PID。

User name
USER_NAME 发布进程的所有者的名称。

Priority
PRIORITY 指示事件的重要程度。不会影响事件分发的顺序。请参阅下表的说明。

I18N catalog
I18N_CATALOG 国际化事件的 I18N 清单文件的名称。

I18N message set id
I18N_SET_ID 标识 I18N 消息清单中的消息集合。

I18N message id
I18N_MSG_ID 事件的 I18N 消息标识。

Format

FORMAT 事件格式文本。有关格式字符串的说明，请参阅 *EvmEvent-Format*(3) 和 *evmtemplate*(4) 参考页。

Reference

REF 对事件说明文本的引用。

事件名称

事件名称是标识事件的主要方式。要发布的事件必须具有事件名称。虽然事件名称可以是任何语法的有效字符串，但该名称通常应标识发布设备并指明结果。

事件名称是 ASCII 字符串，由一系列用点分隔的部分组成，其中最左边的部分表示表示法层次结构的顶部。各部分的子字符串可以包含字母、数字和下划线字符的任意组合。事件名称中可以包含的部分数量没有限制。一个事件模板必须至少包含两个部分。一个事件必须至少包含三个部分才能接受其进行发布。

命名方案提供可扩充的方式来标识事件，允许用户提供任何级别的详细信息。谨慎地命名可使系统管理员直观而准确地选择事件进行查看和监控，并简化系统部分（用于发布事件）的标识和模式（用于对问题发出警告）的识别。事件名称中包含的信息越详细，规范条件就越准确。

事件标识符

事件标识符数据项是数字量，由 EVM 守护程序在发布事件时分配给事件。如果与主机和时间戳数据项一起使用，该值可以用于为事件产生唯一标识。

守护程序根据以下规则分配标识符：

- 每个发布的事件都会接收到一个无符号的整数事件标识号，其值比通过相同 EVM 守护程序发布的前一事件的标识号值大一。
- 在该守护程序启动或重新启动后发布的第一个事件的事件标识符为零。
- 事件标识符在达到其最大值后重新返回为零。
- 如果守护程序接收到的事件已经包含标识符，则不会为该事件生成新的标识符。
- 仅向守护程序验证并接受进行分发的事件分配新的事件标识符。
- 如果接受某一事件并且该事件尚未包含标识符，则即使该事件没有用户，也会向其分配新的标识符。

事件优先级

EVM 仅将事件的优先级值用于过滤、排序和表示 - 而不会将其用于确定分发序列的优先级。优先级是介于 0-700 范围内的整数值，零表示最低优先级。该表指示事件的优先级、evmlogger 针对该优先级采取的缺省操作以及对优先级的说明。

EVM 优先级/名称	缺省值 通知	说明
700 紧急	记录，向超级用户发送邮件	已经检测到危险的情况，并且需要立即进行操作或已经进行操作。
600 - 699 警报	记录，向超级用户发送邮件	即将检测到危险的情况，并且需要立即进行操作或已经进行操作。
500 - 599 关键	记录，向超级用户发送邮件	已经检测到导致系统的某一部分无法运行的故障。
400 - 499 错误	记录	已经检测到系统或应用程序的某一组件中或由某一组件造成的非关键性故障。
300 - 399 警告	记录	需要注意系统或应用程序的某一方面。
200 - 299 注释	记录	通知组件应进行处理的预期操作事件。
100 - 199 信息	无	正常操作事件，例如，某一应用程序已经正常启动或终止。在此范围内的事件通常不会保存在系统 EVM 日志文件中。
1 - 99 调试	无	程序调试信息。此范围内的事件可以进行监控以便获取信息，但通常不会保存在系统 EVM 日志文件中。
0 无	应用程序	优先级 0 应该用于专门由程序预订并且不希望管理员关注的事件。

清单名称和消息集合标识

如果计划对事件进行国际化，需要提供包含适用于所有事件的格式字符串的 I18N 清单文件，并在事件中包含该文件的名称。也可以将文件分解到多个消息集合中，并在事件中提供消息集合标识。但是，请注意与某一特定事件有关的所有消息必须属于同一集合。如果某一单个模板文件中说明的所有事件使用同一清单和/或消息集合中的

消息，可能希望以全局值的形式提供这些项，这样仅须指定它们一次。

设置事件格式以进行显示

虽然事件是不透明二进制结构，但通过使用 **evmshow** 命令或通过调用 **EvmEventFormat()** 函数，可以将其格式设置为用户可读的字符串。格式设置对于面向用户的输出非常重要，但对于仅仅提取事件中包含的任何变量数据，并采取必要操作的应用程序可能是不必要的。

设置事件格式的起始点就是格式数据项。格式就是文本字符串，可以仅包含一段简单的文本、标准数据项的名称，或其值将要替换到文本中的变量的名称，或者以上任意组合。

通过在变量名称前加美元 (**$**) 字符，例如 **$temp** ，可以包含用于替换的变量名称。通过在标准数据项名称前加 **@**，例如 **@timestamp** ，可以包含标准数据项。

EvmEventFormat() 例行程序自动将数据项或变量转换为可显示的形式，而无论其类型为何。

可以在格式文本中转义 **$** 或 **@** 字符的特殊含义，方法是在这些字符前加反斜杠 (\)。要在文本中包含反斜杠，请使用双反斜杠。

如果事件不包含格式数据项，对事件进行格式设置将产生缺省文本字符串，其中包含事件名称以及事件可能包含的任何变量。

如果事件包含重复计数，则重复计数位于输出的前面，格式为 **[***repeat_count***times]** 。

变量数据项

可以在事件中使用变量数据项，以便在每次发布事件时提供不同的数据。

可以为变量赋予任何名称。该名称是与事件一同传送的字符串，用户可以使用该名称以数据的原始格式提取数据。变量名称可以由大写或小写字母数字字符以及下划线字符的任意组合组成。按照惯例，保留以下划线开头的名称以供系统使用。

EVM 的变量数据项具有以下属性：

- 名称
- 类型
- 值
- 大小（对于大多数类型为隐含）
- I18N 消息 id（仅适用于字符串变量）

下表显示 EVM 支持的变量类型：

类型标识	备注
EvmTYPE_BOOLEAN	8 位整数
EvmTYPE_CHAR	8 位字符
EvmTYPE_INT16	16 位有符号整数
EvmTYPE_INT32	32 位有符号整数
EvmTYPE_INT64	64 位有符号整数
EvmTYPE_UINT8	8 位无符号整数
EvmTYPE_UINT16	16 位无符号整数
EvmTYPE_UINT32	32 位无符号整数
EvmTYPE_UINT64	64 位无符号整数
EvmTYPE_FLOAT	32 位浮点数值
EvmTYPE_DOUBLE	64 位浮点数值
EvmTYPE_STRING	以空字符结尾的字符串
EvmTYPE_OPAQUE	二进制数据 - 不能直接解释。必须明确指定大小。

限制

数据类型 **EvmTYPE_FLOAT** 和 **EvmTYPE_DOUBLE** 不能用于在内核中发布的事件。

另请参阅

命令

evmshow(1)。

例行程序

EvmEventFormat(3)。

文件

evmtemplate(4)。

事件管理

EVM(5)。

名称

EvmFilter - EVM（事件管理）事件过滤器

说明

事件过滤器是一组有意义的事件的规范。事件订户使用过滤器将其要接收的事件通知给 EVM 守护进程。例如，一个订户可能只关注接收报告硬件错误的事件，而另一个用户可能希望接收全部高优先级事件（而不管这些事件报告的内容）。如果订户不设置过滤器，则将收不到任何事件。

Event Viewer 和一些 EVM 用户命令还使用过滤器来选择要查看或处理的事件。

过滤器是一个 ASCII 字符串。过滤器可以非常简单，也可以非常复杂。复杂的过滤器可以通过组合简单的过滤器来创建。

简单的过滤器采用以下格式：

[*keyword expr***]** | **all** | **1** | **none** | **0**

expr 的格式特定于过滤器的类型。左侧和右侧方括号（ **[** 和 **]** ）是必需的。可以采用大小写的任何混和形式来指定关键字，其中，下划线字符 (_) 包含在完整长度的关键字（如 **host_name** ）中，可将其省略。关键字可以采用缩写形式，以下各段以大写字母表示每个关键字的最短缩写。

keyword 和关联的 *expr* 的可能值说明如下：

Name *event-name-specifier*

选择名称与 event-name-specifier 匹配的事件。当事件名称与过滤器包含的所有部分相匹配时，该名称才视为是匹配的。

event-name-specifier 可以在任何部分位置中包含 ***** 和 **?** 作为通配符。 ***** 表示具有任何值的 0 个或多个部分。 **?** 精确表示一个部分。任何 *event-name-specifier* 都包含一个隐含的结尾 **.*** 通配符。

Priority *equality-operator integer*

仅传递其优先级符合指定条件的事件。整数值可以为 0 到 700（包括边界值）。有关 equality-operator 的说明，请参阅下表。可将其指定为 **prio** 。

Timestamp *time-range-specifier*

传递其时间戳位于 *time-range-specifier* 内的全部事件。请参阅 *time-range-specifier* 的说明。可将其指定为 **time** 。

Age *equality-operator age-specifier*

选择符合时限规范的事件。请参阅 *age-specifier* 的说明。 *equality-operator* 必须指定 **less-than** 或 **less-than-or-equal** ，表示“早于”，或指定 **greater-than** 或 **greater-than-or-equal** ，表示“晚于”。不允许使用 **equal** 或 **not equal** 运算符。

BEFore *absolute-time-specifier*

传递其时间戳早于 *absolute-time-specifier* 的全部事件。请参阅 *absolute-time-specifier* 的说明。

SINce *absolute-time-specifier*

传递其时间戳等于或晚于 *absolute-time-specifier* 的全部事件。请参阅 *absolute-time-specifier* 的说明。

EVent_id *equality-operator integer*

传递其 *event_id* 符合指定条件的全部事件。有关 *event_id* 的说明，请参阅 *EvmEvent*(5) 。有关 equality-operator 的说明，请参阅下表。 **EVent_id** 关键字可能会缩写为 **ID** 。

NONE 或 **0**　　过滤器值为 NONE 或 0（零）则不会传递任何事件。

ALL 或 **1**　　过滤器值为 ALL 或 1 将传递全部事件。

下表显示了可用的 *equality-operator* 说明符及其备用形式。可以大小写的任何混和形式使用备用形式。

运算符	备用形式	含义
=	**eq**	等于
>	**gt**	大于
<	**lt**	小于
>=	**ge**	大于或等于
<=	**le**	小于或等于
!=	**ne**	不等于

age-specifier 由整数值及紧随的一个或多个字母 **w** （周）、 **d** （天）、 **h** （小时）、 **m** （分）或 **s** （秒）组成。age-specifier 会生成相对于当前时间的绝对时间值，通过 **evmget** 或 Event Viewer 检索历史事件时此值可能会很有用。设置 EVM 日志程序或 evmwatch 使用的过滤器时，使用 *age-specifier* 是没有意义的。

如果指定了数周的时间，则此时间乘以 7 会转换为天数。如果要计算以周或天指定的时限的绝对时间，则第一天始终被视为是前一天的午夜到现在的时间，在此之前的天数是以午夜到午夜的方式计算的。例如，如果指定 *age-specifier* 为 **1d** ，则相对于同一天上午 12:00 来选择事件。值 **2d** 将选择相对于上一天上午 12:00 来选择事件。值 **0d** 有效，它等效于 **1d** 。有关详细信息，请参阅下面的示例。

如果指定了数小时、数分或数秒的期间，则可以从当前时间减去时限来计算绝对时间，而不考虑时间的边界值。例如，如果在 **15:23:14** 指定 *age-specifier* 为 **24h** ，则相对于上一天的 15:23:14 来选择事件。

time-range-specifier 包含七个以冒号分隔的字段，其格式如下：

year:*month-of-year*:*day-of-month*:*day-of-week*:*hours*:*minutes*:*seconds*

时间范围中的任何部分都可以由作为通配符的星号 (*) 字符替换，表示此部分中的任何值都将与过滤器相匹配。可以通过逗号将值隔开，为各部分指定多个离散值。可以使用连字符分隔范围的开始值和结束值，以便指定范围。 *absolute-time-specifier* 与 *time-range-specifier* 非常相似。它具有六个部分，并且不允许使用通配符。其格式如下：

year:*month-of-year*:*day-of-month*:*hours*:*minutes*:*seconds*

下表显示了时间规范、各部分的值的范围两种形式。

说明符	范围
year	1970 到 2030
month-of-year	1 到 12
day-of-month	1 到 31
day-of-week	0（星期日）到 6
hours	0 到 23
minutes	0 到 59
seconds	0 到 59

通过使用 NOT 运算符、即感叹号 (**!**) 或关键字 **NOT** ，可以反转（逻辑反转）任何表达式。

通过使用 AND （ & 或关键字 **AND** ）和 OR （ | 或关键字 **OR** ）逻辑运算符，可以将两个或多个简单的过滤器组成复杂的过滤器。可以使用括号（ **(** 和 **)** ）对部分过滤器表达式分组，以设置测试操作的优先级。逻辑运算符和分组运算符的优先级顺序（从最高到最低）是：

( ) ! & |

事件过滤器可以是直接过滤器，也可以是间接过滤器。直接过滤器是显示在过滤器规范位置的文本字符串。间接过滤器包含在文件中，且使用以下语法进行引用：

@filename:filtername

有关使用间接过滤器的详细信息，请参阅 *evmfilterfile*(4) 。

如果正在求值的事件不包含过滤器表达式中正在比较的项目，则表达式始终不会生成任何匹配项。例如，如果事件中缺少 *timestamp* 项目，且在过滤器字符串中包含了 before 关键字，则过滤器的该部分将不返回匹配项。

注释

EVM 的后续版本可能会通过添加新关键字或运算符来改进过滤器语法。

举例

下表显示了多个过滤器规范，以及对每个过滤器规范的解释。

过滤器字符串	解释
"[name *]"	任何命名事件。
"[name myco.*]"	名称以 **myco** 开头的所有事件。
"![name myco.*]"	名称不以 **myco** 开头的所有事件。
"[name ?.?.?]"	名称至少包含三部分的任何事件。
"[name myco.myapp.*]"	名称的前两个部分为 **myco.myapp** 的任何事件。

"[name myco.myapp]" 名称的前两个部分为 **myco.myapp** 的任何事件。与上一个过滤器字符串含义相同。

"[name sys.unix.syslog]" 名称的前三个部分为 **sys.unix.syslog** 的事件。

"[name myco.myapp.*.showme]" 以部分 **myco.myapp** 开头和以 **showme** 结尾的任何事件名称，不论两者之间包含多少个部分。

"[age < 1d]" 今天发布的任何事件。

"[age < 4w]" 在上 4 个星期内发布的任何事件。

"[age lt 30s]" 上 30 秒内发布的任何事件。

"[age gt 1d]" 昨天发布的任何事件。

"[time 2000:6:1:*:*:*:*]" 2000 年 6 月 1 日发布的任何事件。

"[time 2000:6:1,3:*:*:*:*]" 2000 年 6 月 1 日 或 6 月 3 日发布的任何事件。

"[time 2000:6:1-3:*:*:*:*]" 2000 年 6 月 1 日至 6 月 3 日之间发布的任何事件。

"[time 2000:6:1-3,5-7:*:*:*:*]" 2000 年 6 月 1 日至 6 月 3 日之间（包括边界值）发布的任何事件，或 2000 年 6 月 5 日至 6 月 7 日之间（包括边界值）发布的任何事件。

"[time *:*:*:*:00-02:*:*]" 在午夜至第二天上午 2:59:59 之间（包括边界值）发生的所有事件。

"[since 2000:6:1:03:00:00]" 2000 年 6 月 1 日上午 3:00 之后发生的所有事件。

"[before 2000:6:1:03:00:00]" 2000 年 6 月 1 日上午 3:00 之前发生的所有事件。

"[prio > 500]" 优先级高于 500 的所有事件

"[name myco.myapp] & [pri >= 500]" 名称以 myco.myapp 开头且优先级至少为 500 的所有事件。

"[name myco.myapp] | [pri >= 500]" 名称以 myco.myapp 开头或优先级至少为 500 的所有事件。

"[name sys.unix.evm] & [age < 2d]" 今天或昨天发生的所有 evm 事件。

"[name sys.unix.evm] and [time 2000:6:1-3:*:*:*:*]"
2000 年 6 月 1 日、2 日或 3 日发生的所有 evm 事件。

"none" 不允许传递任何事件。

"0" 不允许传递任何事件。

"all" 传递全部事件。

"1" 传递全部事件。

"@sys" 指定间接过滤器。此过滤器字符串是名为 **sys** 或 **sys.evf** 的过滤器文件中包含的缺省过滤器。

"@sys:evm" 指定间接过滤器。此过滤器字符串是一个名为 **evm** 的过滤器，它包含在名为 **sys** 或 **sys.evf** 的过滤器文件中。

另请参阅

命令

evmget(1)、 evmshow(1) 和 evmwatch(1)。

例行程序

EvmConnSubscribe(3) 、 EvmFilterCreate(3) 、 EvmFilterDestroy(3) 、 EvmFilterIsFile(3) 、 EvmFilterReadFile(3) 、 EvmFilterSet(3) 和 EvmFilterTest(3)。

文件

evmfilterfile(4)。

事件管理

EVM(5)。

EVM 事件

EvmEvent(5)。

名称

executable_stack - 控制缺省情况下程序堆栈是否可执行

值

无故障

1

缺省值

0

允许值

0-2

建议值

0-2

说明

此可调参数控制缺省情况下程序堆栈是否可执行。它允许对系统进行配置，使其受到额外保护，以免遭受缓冲区溢出攻击而不会降低系统性能。这类攻击通常尝试欺骗特权程序来执行未授权的操作或进行未授权的访问。在网页上搜索“Smashing the Stack for Fun and Profit”可以获得这类攻击的背景信息。

HP-UX 上运行的大多数程序都不需要执行位于其堆栈的代码。少数程序，特别是有些模拟程序、解释程序和较旧版本的 Java，可能有合法的理由来从其堆栈执行代码。这些程序通常都有自修改代码。通过使用此可调参数和 **chatr** 命令的 **+es** 选项的组合，这些可执行文件在执行同时仍会对系统的其他部分进行保护。

有关详细信息，请在更改此可调参数之前参考 *chatr*(1) 联机帮助页“限制对堆栈的执行权限”部分。

应该由谁来更改此可调参数？

任何用户。

对于更改的限制

对此可调参数的更改在更改之后新进程启动时生效。

何时应更改此可调参数的值？

此可调参数控制操作模式而不是数据结构的大小和限制。对一个系统的适当设置取决于用户考虑安全性最重要还是兼容性最重要。

值为 **1** 与先前版本的 HP-UX 兼容，但是它的安全性最差。此设置允许执行位于程序堆栈上的潜在的恶意代码。

值为 **2** 提供了有关任何程序尝试在其堆栈上执行代码的警告，但是不会更改程序的行为。可疑的活动被记录到内核消息缓冲区中。（请参阅 *dmesg*(1M)）。这旨在允许用户安全地确定可调参数值为 **0** 是否会影响合法应用程序的“试验模式”的设置。

对于较高级别的安全性很重要的系统，推荐配置可调参数值为 **0** 。这实际上与设置为 **2** 相同，但是它还会终止任何尝试执行堆栈上的代码的进程。该进程会在潜在的非法代码执行前被终止。

更改此值的副作用是什么

除非应用程序尝试执行位于其堆栈上的指令，否则此可调参数不会影响系统行为。大部分编写的 HP-UX 应用程序都不会进行此操作。

同时还应更改哪些其他可调参数的值?

无。

警告

所有 HP-UX 内核可调参数都是针对于特定版本的。将来的 HP-UX 版本中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅网站 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

executable_stack 由 HP 开发。

名称

expanded_node_host_names - 对系统节点名和主机名启用最大长度的扩展

值

保证安全

0 - 禁用

缺省值

0 - 禁用

允许值

0 - 禁用

1 - 启用

说明

expanded_node_host_names 可调参数控制系统节点名和主机名的长度。如果此可调参数值为 0，则用于设置节点名和主机名的系统实用程序允许兼容的名称长度。也就是说，对于节点名和主机名，分别为 8 字节和 64 字节。如果此可调参数值为 1，则实用程序最多允许 255 字节的扩展长度。

更改此可调参数不影响当前的系统节点名和主机名。

如果节点名和主机名的长度分别大于 8 字节和 64 字节，则使用这些名称的应用程序可能会出现异常或错误的行为。有关详细信息，请参阅“警告”一节。

更改此可调参数的人员

如果需要更改系统节点名和（或）主机名允许的最大长度，系统管理员可以更改此可调参数。

更改限制

此可调参数为动态的。将其值更改为 1 后，管理员可立即分配扩展的节点名或主机名（最多 255 字节）。如果将其值更改为 0，则对于以后进行的节点和主机的名称设置将强制最大长度为 8 字节和 64 字节，但不影响当前的名称设置。

应该何时增加此可调参数的值

如果需要分配扩展的节点名和（或）主机名，则可以将此可调参数增加到 1。

增加此值的负面影响

如果将可调参数值增加到 1，则用于设置节点名和主机名的系统实用程序允许最大长度为 255 字节的名称。

应该何时降低此可调参数的值

如果确定不需要使用扩展的节点名和主机名，则应该将此可调参数降低到 0。

降低此值的负面影响

如果将可调参数值降低到 0，则用于设置节点名和主机名的系统实用程序允许最大长度分别为 8 字节和 64 字节的名称。

应该同时更改的其他可调参数值

无。

实际应用信息

有关应用的详细信息，请参阅白皮书《Node and Host Name Sizes on HP-UX: Using the Expanded Capabilities》。

警告

如果节点名和主机名的长度分别超过 8 字节和 64 字节，则使用这些名称的应用程序可能会出现异常行为或失败。管理员应该阅读并理解相关文档中说明的问题（请参阅白皮书以及 *nodehostnamesize*(5)）。

更改此可调参数时，可调参数处理程序始终会发出警告。

如果将此可调参数值设置为 0（禁用），用于设置节点名和主机名的系统接口及命令将以无提示方式，将输入分别截断为 8 字节或 64 字节。该行为与以前的 **HP-UX** 版本一致。此操作不提供错误，也不提供警告。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

作者

expanded_node_host_names 由 HP 开发。

另请参阅

hostname(1)、uname(1)、setuname(1M)、gethostname(2)、sethostname(2)、uname(2)、hostname(5)、nodehostnamesize(5)。

《Node and Host Name Sizes on HP-UX: Using the Expanded Capabilities》白皮书，可从 **http://docs.hp.com** 上获得。

名称

fadvise： fadvise.h - 使用 fadvise() 函数时所需的结构

概要

```
#include <sys/fcntl.h>
#include <sys/fadvise.h>
```

说明

<sys/fadvise.h> 头文件定义使用 **fadvise()** 函数（请参阅 *fadvise*(2) ）时所需的 **fad_extparms** 和 **fad_parms** 结构及其他文字。

fad_extparms 结构包含下列成员：

int	***fad_count***	**fad_plist[]** 中的条目数
struct fad_parms	***fad_plist[1]***	struct **fad_parms** 的数组

fad_parms 结构包含下列成员：

enum fadv_hints	***fad_advice***	适用于参数元组的提示。
off_t	***_fad_res1***	可以是下列其中一项：文件中范围的字节偏移量； cc-numa 系统的内存分配策略。
size_t	***_fad_res2***	可以是下列其中一项：文件中范围的字节大小；在 cc-numa 系统上分配内存时使用的内存位置；将脏页写入文件系统之前需要经过的时间。

实际应用信息

定义了下列宏和别名，以便与 ISO POSIX-1 标准保持一致。

```
#ifdef _APP32_64BIT_OFF_T
#define posix_fadvise(FD, OFF, LEN, CMD) fadvise64(FD, OFF, LEN, CMD, NULL)
#else
#define posix_fadvise(FD, OFF, LEN, CMD) fadvise(FD, OFF, LEN, CMD, NULL)
#endif
```

```
#define POSIX_FADV_NORMAL      FADV_NORMAL
#define POSIX_FADV_SEQUENTIAL  FADV_SEQUENTIAL
#define POSIX_FADV_RANDOM      FADV_RANDOM
#define POSIX_FADV_WILLNEED    FADV_WILLNEED
#define POSIX_FADV_DONTNEED    FADV_DONTNEED
#define POSIX_FADV_NOREUSE     FADV_NOREUSE
```

可以将下列别名与 **FADV_CCNUMA** 提示一起使用。

```
#define FAD_POLICY          _fad_res1 /* allocation policy */
```

可以将下列别名与 **FADV_LARGEPAGE_HINT** 提示一起使用。

#define FAD_PGSIZE _fad_res2 /* preferred large page size */

可以将下列别名与 **FADV_SYNC_RDWR** 提示一起使用。

#define FAD_SYNC_TIME _fad_res2 /* pageout interval in secs. */

举例

```
#include <sys/fcntl.h>
#include <sys/fadvise.h>

fad_extparms_t fad_extparms;

main(int argc, char *argv[])
{

  int fd;

  fd = open(argv[1], O_RDONLY);

  /* To setup a CCNuma policy of MEM_FIRST_TOUCH */
  bzero(&fad_extparms, sizeof(fad_extparms_t));
  fad_extparms.fad_count = 1;
  fad_extparms.fad_plist[0].fad_advice      = FADV_CCNUMA;
  fad_extparms.fad_plist[0].FAD_POLICY      = FADV_VM_MEM_FIRST_TOUCH;
  fadvise(fd, 0, 0, 0, &fad_extparms);
}
```

警告

对于 32 位应用程序，在使用 64 位值的文件系统中， **_fad_res1** 将被截断成最低 32 位有效位形式。

另请参阅

fadvise(2)、 fcntl(2)、 posix_fadvise(2)、 fcntl(5)、 types(5)。

符合的标准

<sys/fcntl.h>： POSIX.1

名称

fcache_fb_policy - 用于 VxFS 文件系统的后台刷新请求的策略

值

保证安全

0

缺省值

0

允许值

最小值： **0**

最大值： **1**

说明

VxFS 文件系统具有一项称为“后台刷新”的功能，该功能使用异步 I/O 刷新文件的脏页。当内存中文件的脏页数达到某一特定阈值时，通常会执行此操作。这些刷新操作通过各种 VxFS 可调参数来控制，例如：**write_pref_io** 、 **max_diskq** 、 **write_throttle** 和 **write_nstream** 。有关 VxFS 可调参数的详细信息，请参阅 *vxtunefs*(1M) 联机帮助页。

后台刷新功能有助于降低与文件关联的脏页数。该功能还可提高数据的完整性，并在系统崩溃时避免发生数据损坏。但是，该后台刷新功能的缺点是，从线程环境启动 I/O 的开销过高。此外，由于上一后台刷新操作，可能已经启动对数据块执行的 I/O，因此，对该数据块的重新写入操作可能会暂停。

HP-UX 提供了一个 Syncer 守护程序，可最大限度地满足后台刷新功能的要求。可将该守护程序调整为大约每 30 秒运行一次，以便刷新脏页，从而保证文件系统的完整性。有关 **syncer** 守护程序的详细信息，请参阅 *syncer*(1M) 联机帮助页。在 HP-UX 11i V3 上可禁用 VxFS 后台刷新功能，并在系统崩溃时依靠 syncer 守护程序确保数据的可用性和完整性，这有助于提供最佳性能。

fcache_fb_policy 可调参数可用于在写入性能和数据可用性之间进行权衡。 **fcache_fb_policy** 可调参数具有下列两种设置：

fcache_fb_policy = 0
: 该设置为高性能设置，没有实现后台刷新功能。应用程序将依赖于 syncer 守护程序来提供文件数据的完整性。或者，如果应用程序要减少系统上的脏页数，则必须通过 *fsync*(2) 执行显式刷新。

fcache_fb_policy = 1
: 该设置与 HP-UX 11i v2 的缺省行为相对应。该设置可启用文件系统的后台刷新功能，旨在最大程度地减少文件的脏页。其负面影响为，如果在 syncer 守护程序将脏页刷新到磁盘之前发生系统故障，则可使数据丢失的可能性降到最低。

更改此可调参数的人员

负责运行执行大文件写入操作的应用程序的系统管理员可更改此可调参数。

更改限制

对此可调参数的更改将立即生效。

何时应更改此可调参数的值

如果系统上运行的应用程序所需的数据完整性要求高于 HP-UX syncer 守护程序所提供的数据完整性，则应将此可调参数从 0 更改为 1。

此外，如果设置了 VxFS 可调参数 **write_throttle** ，则还应将 **fcache_fb_policy** 更改为 1。 **write_throttle** 可调参数允许管理员在将文件系统生成的每个文件的脏缓冲区写入到磁盘之前，降低该缓冲区的数量。 **write_throttle** 的缺省值为零。该缺省值对每个文件的脏缓冲区的数量没有任何限制。通过 **write_throttle** 缺省设置，绕过后台刷新功能不会发生任何问题。但是，如果启用了 **write_throttle** ，执行写入操作的应用程序可能会延迟，这是因为它们在继续运行之前将等待 **syncer** 刷新某些脏缓冲区。

更改此值的负面影响

将该值从 0 更改为 1 将导致 I/O 活动增加，还可能丧失整个 *write*(2) 性能。

应该同时更改的其他可调参数

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

fcache_fb_policy 由 HP 开发。

另请参阅

syncer(1M)、 vxtunefs(1M)、 fsync(2)、 write(2)。

名称

fcache_seqlimit_file - 顺序访问可以消耗的文件缓存百分比，每个文件限制

值

保证安全

100

缺省值

100

允许值

允许的最小值为 **0** 。允许的最大值为 **100** 。

指定一个正整数值。

说明

此参数对顺序访问可以在文件缓存中消耗的内存量设置每个文件限制。请务必注意此可调参数与系统级限制可调参数 **fcache_seqlimit_system** 之间的关系。结合使用这两个可调参数能够控制顺序访问可以在文件缓存中消耗的内存量。

如果对文件的顺序访问超出了每个文件限制以及系统级限制，则移出多余的文件缓存页。请注意，要发生页面窃取，必须达到这两个限制。例如，如果未达到 **fcache_seqlimit_system** 限制，文件可能会超过其 **fcache_seqlimit_file** 限制。同样，如果系统上的文件都没有超过每个文件限制，那么，即使达到了系统级限制，也不发生页面窃取。页面窃取功能专门用于提升大文件的顺序 I/O 性能。

对文件缓存内存的消耗进行限制可以缓解文件缓存中的内存压力。还可以防止缓存擦除 - 单个线程按顺序访问大文件会擦除文件缓存的现有内容。

更改此可调参数的人员

运行需要执行较大顺序文件 I/O 的应用程序的系统管理员。通常，这些文件大于系统中物理内存的大小。

更改限制

对此可调参数的更改立即生效。

应该何时增加此可调参数的值

由于顺序访问，希望允许更大的文件缓存消耗时。将 **fcache_seqlimit_system** 和 **fcache_seqlimit_file** 设置为 100 可以有效地禁用顺序访问页面窃取功能。

增加此值的负面影响

将此可调参数设置得过高可能使大文件（大于物理内存大小）的大顺序 I/O 性能下降。应该针对预期的工作负荷相应调整 **fcache_seqlimit_system** 和 **fcache_seqlimit_file** 。

应该何时降低此可调参数的值

由于顺序访问，希望限制文件缓存消耗时。将 **fcache_seqlimit_system** 和 **fcache_seqlimit_file** 设置为 0 可以强制顺序访问页面窃取始终发生。

降低此值的负面影响

尝试限制文件缓存消耗时，可能会发生更多的页面移出。

应该同时更改的其他可调参数值

应同时更改 **fcache_seqlimit_system** 和 **fcache_seqlimit_file** 。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

fcache_seqlimit_file 由 HP 开发。

另请参阅

fcache_seqlimit_system(5)。

名称

fcache_seqlimit_system - 顺序访问可以消耗的文件缓存百分比，每个系统级限制

值

保证安全

100

缺省值

100

允许值

允许的最小值为 **0** 。允许的最大值为 **100** 。

指定一个正整数值。

说明

此参数对顺序访问可以在文件缓存中消耗的内存量设置每个系统级限制。请务必注意此可调参数与按文件限制可调参数 **fcache_seqlimit_file** 之间的关系。结合使用这两个可调参数能够控制顺序访问可以在文件缓存中消耗的内存量。

如果对文件的顺序访问超出了每个文件限制以及系统级限制，则移出多余的文件缓存页。请注意，要发生页面窃取，必须达到这两个限制。例如，如果未达到 **fcache_seqlimit_system** 限制，文件可能会超过其 **fcache_seqlimit_file** 限制。同样，如果系统上的文件都没有超过每个文件限制，那么，即使达到了系统级限制，也不发生页面窃取。

页面窃取功能专门用于提升大文件的顺序 I/O 性能。对文件缓存内存的消耗进行限制可以缓解文件缓存中的内存压力。还可以防止缓存擦除 - 单个线程按顺序访问大文件会擦除文件缓存的现有内容。

更改此可调参数的人员

运行需要执行较大顺序文件 I/O 的应用程序的系统管理员。通常，这些文件大于系统中物理内存的大小。

更改限制

对此可调参数的更改立即生效。

应该何时增加此可调参数的值

由于顺序访问，希望允许更大的文件缓存消耗时。将 **fcache_seqlimit_system** 和 **fcache_seqlimit_file** 设置为 100 可以有效地禁用顺序访问页面窃取功能。

增加此值的负面影响

将此可调参数设置得过高可能使大文件（大于物理内存大小）的大顺序 I/O 性能下降。应该针对预期的工作负荷相应调整 **fcache_seqlimit_system** 和 **fcache_seqlimit_file** 。

应该何时降低此可调参数的值

由于顺序访问，希望限制文件缓存消耗时。将 **fcache_seqlimit_system** 和 **fcache_seqlimit_file** 设置为 0 可以强制顺序访问页面窃取始终发生。

降低此值的负面影响

尝试限制文件缓存消耗时，可能会发生更多的页面移出。

应该同时更改的其他可调参数值

应同时更改 **fcache_seqlimit_system** 和 **fcache_seqlimit_file** 。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

fcache_seqlimit_system 由 HP 开发。

另请参阅

fcache_seqlimit_file(5)。

名称

fcntl - 文件控制选项

概要

#include <sys/types.h>

#include <fcntl.h>

说明

fcntl() 函数为控制打开文件而提供。 **<fcntl.h>** 包括描述了 **fcntl()** 和 **open()** 的 请求和 参数的文件。请参阅 *fcntl*(2) 和 *open*(2) 。

通过 **open()** 设置并由 **fcntl()** 访问的访问模式包括:

O_RDONLY
O_WRONLY
O_RDWR

文件访问模式的掩码为:

O_ACCMODE

通过 **open()** 或 **fcntl()** 设置并通过 **fcntl()** 访问的文件状态标记包括:

O_NDELAY	非阻塞 I/O。
O_NONBLOCK	POSIX 样式的非阻塞 I/O。
O_APPEND	追加（保证在末尾写入）。
O_DSYNC	数据通过缓存写入。
O_SYNC	数据和属性通过缓存写入。
O_RSYNC|O_DSYNC	读取和写入时数据通过缓存写入。
O_RSYNC|O_SYNC	读取和写入时数据和属性通过缓存写入。
O_LARGEFILE	当文件系统挂接为启动的大文件时， **O_LARGEFILE** 选项允许该文件大于 2 GB。

标记 **O_SYNCIO** 与 **O_SYNC** 同义，并且定义用于在 **<fcntl.h>** 中向后兼容。

仅 **open()** 可访问的标记值有:

O_CREAT	打开同时创建文件（使用第三方打开参数）。
O_TRUNC	打开的同时进行截断。
O_EXCL	独占打开。
O_NOCTTY	不分配控制终端。
O_NOFOLLOW	不遍历作为路径最后一部分的符号链接。

对 **fcntl()** 的请求包括:

F_DUPFD	复制文件描述符。
F_GETFD	获得文件描述符标记。
F_SETFD	设置文件描述符标记。
F_GETFL	获得文件标记。
F_SETFL	设置文件标记。
F_GETLK	获得阻塞文件锁。
F_SETLK	设置或清除文件锁并在忙时失败。
F_SETLKW	设置或清除文件锁并在忙时等待。
F_ADVISE	Fadvise 服务请求（请参阅 *fadvise*(2)）。

F_GETFD 、 **F_SETFD** 的文件描述符标记为:

FD_CLOEXEC

文件段锁定控制结构 **struct flock** ，包括下列成员:

```
short l_type;      /* F_RDLCK, F_WRLCK or F_UNLCK */
short l_whence;    /* Flag - see lseek(2) */
off_t l_start;     /* Relative offset in bytes */
off_t l_len;       /* Size; if 0 then until EOF */
pid_t l_pid;       /* By F_GETLK - process holding lock */
```

文件段锁定类型有:

F_RDLCK	**Read lock.**
F_WRLCK	**Write lock.**
F_UNLCK	**Remove locks.**

另请参阅

fcntl(2)、fadvise(2)、open(2)。

符合的标准

<fcntl.h>：AES、SVID3、XPG2、XPG3、XPG4、FIPS 151-2、POSIX.1

名称

fenv - 浮点环境宏和函数

概要

#include <fenv.h>

说明

头文件 **<fenv.h>** 声明了两种类型和多个宏以及函数，以提供对浮点环境的访问。 浮点环境是对浮点状态标记和浮点控制模式的统称。

浮点状态标记是一个系统变量，其值在引发浮点异常时设置(但决不能清除），该异常是异常的浮点运算产生的负面影响，用于提供辅助信息。 浮点控制模式是一个系统变量，其值可以由用户设置来影响后续浮点运算的行为；在 HP 9000 和 HP Integrity 服务器上，控制模式包括舍入方向模式、渐进/下溢到零的下溢模式以及陷阱启用。

定义了下列类型：

fenv_t	表示整个浮点环境。
fexcept_t	表示所有浮点异常标记。

下列宏表示浮点状态标记。它们定义为具有值的整型常量表达式，这样所有宏组合的按位 OR 运算将产生不同的值。

FE_INEXACT	不精确异常。
FE_DIVBYZERO	除数为零异常。
FE_UNDERFLOW	下溢异常。
FE_OVERFLOW	上溢异常。
FE_INVALID	无效操作异常。
FE_ALL_EXCEPT	所有异常宏的按位 OR 运算。

下列宏表示舍入方向模式。它们定义为具有不同非负值的整数常量表达式。

FE_TONEAREST	舍入到最近舍入方向模式。
FE_UPWARD	舍入到正无穷舍入方向模式。
FE_DOWNWARD	舍入到负无穷舍入方向模式。
FE_TOWARDZERO	舍入到零舍入方向模式。

下列宏定义为指向常量限定的 **fenv_t** 的指针：

FE_DFL_ENV	缺省浮点环境。

对 ISO/IEC C99 指定的设备，HP 实现方案增加了四种特定于 HP 的函数： **fegetflushtozero()** 、 **fesetflushto-**

zero()、 **fegettrapenable()** 和 **fesettrapenable()** 。

文件

/usr/include/fenv.h

另请参阅

feclearexcept(3M) 、 fegetexceptflag(3M) 、 feraiseexcept(3M) 、 fesetexceptflag(3M) 、 fetestexcept(3M) 、 fegetround(3M) 、 fesetround(3M) 、 fegetenv(3M) 、 feholdexcept(3M) 、 fesetenv(3M) 、 feupdateenv(3M) 、 fegetflushtozero(3M)、 fesetflushtozero(3M)、 fegettrapenable(3M)、 fesettrapenable(3M)、 math(5)。

符合的标准

<fenv.h> ： ISO/IEC C99（包括附件 F：“IEC 60559 floating-point arithmetic”）

名称

filecache_max、filecache_min - 用于缓存文件 I/O 数据的最大或最小物理内存量

值

保证安全

filecache_min：大约为物理内存的 **5%**

filecache_max：大约为物理内存的 **5%**

缺省值

filecache_min：大约为物理内存的 **5%**

filecache_max：大约为物理内存的 **50%**

根据系统上的物理内存量自动计算和调整缺省值。

由 **kctune** 显示时，当前缺省值将表示为系统在内部使用的当前值（字节表示）。显示的值会随着内部值的自动调整而变化。

允许值

Minimum **filecache_min**： 不小于 1 MB 的等效值

Minimum **filecache_max**： 不小于 1 MB 的等效值

Maximum **filecache_min**： 不大于物理内存的 70% 的等效值

Maximum **filecache_max**：不大于物理内存的 90% 的等效值

请参阅下面的 更改限制。

可以按如下所示指定值：

1) 缺省值

2) 总物理内存的百分比：
正整数后跟百分比符号（例如，70%）。

3) 常量值：
一个表示物理内存字节数的正整数，其后可以紧跟一个乘数后缀，其中 **K=1000** 、 **KB=1024** 、 **M=1000000** 、 **MB=(1024*1024)** 、 **GB=(1024*1024*1024)** 。

建议值

建议使这些可调参数处于自动（缺省）状态，以便系统能够更好地平衡文件系统 I/O 密集型进程以及其他类型的进程之间的内存使用量。

说明

这些可调参数控制文件系统 I/O 操作过程中可用于缓存文件数据的物理内存量。

由 **filecache_min** 可调参数指定的物理内存量将被保留并确保可用于文件缓存。

用于文件缓存的物理内存量的范围介于 **filecache_min** 与 **filecache_max** 之间，具体取决于 I/O 负载以及对物理内存的争用请求。

如果将这些可调参数设置为缺省值或百分比值，则它们将相应地根据联机添加或删除 (OL*) 的物理内存进行自动调整。

更改此可调参数的人员

自动（缺省）状态应适合于大多数环境。

如果要指定粒度比物理内存百分比更细的文件缓存限制，必须将这些可调参数设置为常量值而不是缺省值或百分比（例如，设置一个小于物理内存的 1% 的最小大小或固定大小）。

如果不希望文件缓存限制根据物理内存的 OL* 而调整，则必须将这些可调参数设置为常量值（而不是缺省值或百分比）。

要优先考虑执行大型文件 I/O 活动的系统上起决定作用的 I/O，或者相反，要优先考虑非 I/O 密集型进程的更佳性能，则可以考虑更改这些可调参数的值，同时应注意如上所述的负面影响。

要确定一个合理的缓存大小的值，应考虑系统上的文件 I/O 密集型应用程序及其工作集的大小。根据应用程序的类型，工作集大小可能基于事务的大小，或给定时间单位内的数据大小。例如，要确定一个保守的 **filecache_min** 值（以 MB 为单位），可以使用以下公式：

> **MB =** *number-of-system-processes* ***** *largest-application-record-length-in-MB* ***** *number-of-records-in-working-set*

只有那些主动使用磁盘 I/O 处理文件数据的进程才应包含在计算内。其他进程可以排除。以下示例表明哪些进程应该包含在计算中、哪些进程应该从计算中排除。

包含：NFS 守护程序、文本格式设置程序、数据库管理应用程序、文本编辑器、编译程序等，这类程序访问或使用存储在一个或多个挂接在系统上的文件系统中的源文件和（或）输出文件。

排除：X 显示应用程序、 **hpterm** 、 **rlogin** 、登录 Shell、系统守护程序、 **telnet** 或 **uucp** 连接等。这些进程会使用很少（如果有的话）的磁盘 I/O 用于文件数据。

更改限制

这些可调参数具备动态性和自动性。

系统将指定的可调参数值向下舍入到最近的物理页边界。

由 **filecache_min** 表示的物理内存量必须等于或小于由可调参数 **filecache_max** 表示的内存。

将这些可调参数设置为常量值将取消它们与 OL* 事件的耦合。

如果没有足够的可用物理内存用于满足请求，则调大 **filecache_min** 参数的值可能失败。

这些可调参数（ **filecache_min** 和 **filecache_max** ）必须 同时设置为相对状态（缺省状态或百分比状态），或必须同时设置为常量值。例如，以下是可接受的设置：

kctune filecache_min=100MB filecache_max=100MB

kctune filecache_min=15% filecache_max=65%

kctune filecache_min=Default filecache_max=65%

以下设置将导致错误:

kctune filecache_min=10% filecache_max=1GB

ERROR:The specified units for the values of filecache_min and filecache must be consistent:both as relative (percentage or default) or both as constant value.

如果 **filecache_min** 当前设置为缺省值，则以下是可以接受的设置:

kctune filecache_max=Default

kctune filecache_max=40%

但以下设置将导致错误:

kctune filecache_max=1GB

请参阅上面 允许值一节中的其他限制。

应该何时增加这些可调参数值

如果在初始化时以及（或）在包含文件系统 I/O 密集型进程的系统上，系统性能较低，则可能表示这些可调参数的值设置得过低。如果有大量进程不断地使用文件数据 I/O，则应针对更具决定性的 I/O 增加 **filecache_min** 的值。在大多数情况下，尤其是当文件数据 I/O 只应偶尔出现高峰时，建议增加最大限制 **filecache_max** 的值。

增加这些值的负面影响

为最小文件缓存大小（由 **filecache_min** 表示）保留的内存量不能在系统上用于其他用途。请注意不要将该值增加得过大，否则最终会产生内存压力并降低总体系统性能。

应该何时降低这些可调参数值

根据争用请求的不同，可以降低最小限制 **filecache_min** 的值，以便将更大的内存百分比用于文件系统 I/O 缓存以外的用途。通过降低 **filecache_max** 的值，将有更大量的内存可用于其他用途，而不会与文件 I/O 请求进行竞争。

降低这些值的负面影响

如果有很多物理内存争用请求，并且文件缓存可调参数设置的值过低，则对文件 I/O 操作极高的要求最终会导致文件系统 I/O 性能降低。

举例

将文件缓存最小值设置为物理内存的 10%:

kctune filecache_min=10%

设置一个固定大小为 1 GB 的文件缓存:

kctune filecache_min=1GB filecache_max=1GB

将文件缓存最小值设置为物理内存的 15%，并将最大值设置为物理内存的 65%：

kctune filecache_min=15% filecache_max=65%

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除这些参数，或改变其含义。

先前 HP-UX 版本中存在的其他与调整缓冲区缓存大小相关的可调参数现在已过时。应使用可调参数 **filecache_min** 和 **filecache_max** 设置文件缓存限制。请注意，在任何给定系统上，这两个新可调参数的最佳值并不一定等于较早系统中已过时的可调参数的最佳值。在尝试更改新文件缓存可调参数的值时，应首先确定新缺省值是否会产生可接受的性能。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

filecache_max 和 **filecache_min** 由 HP 开发。

另请参阅

kctune(1M)、 sam(1M)、 gettune(2)、 settune(2)。

名称

fs_async - 在写入操作完成前启用写入调用来返回

值

保证安全

0

缺省值

0

允许值

允许值包括：

0 （仅使用同步磁盘写入），或者

1 （允许异步磁盘写入）。

指定一个正整数值 **0** 或者 **1** 。

说明

fs_async 指定是否允许文件系统数据结构对磁盘进行异步写入操作。

当文件系统数据结构在文件系统上进行更新时，如果发生了系统崩溃，则同步写入磁盘将可以更容易地恢复文件系统完整性。

如果选择异步写入，则将保留 NFS 群集环境的 HP-UX 文件系统语义。此外，如果已将同步写入功能配置到内核中，则将继续同步写入使用标记为 **0_SYNC** 的 **open()** 打开的文件。

对磁盘进行异步写入操作可以显著地改善系统性能。但是，如果发生系统崩溃，则异步写入会使文件系统数据结构处于不一致的状态。有关何时选择同步或异步写操作的详细信息，请参阅下列教程。

更改限制

此可调参数是静态的。对此可调参数值的任何更改都将在生效前要求系统重新引导。

教程：什么是同步写入和异步写入？

如果一个文件打开进行写入，并且正在将数据写入该文件中，则数据将在缓冲区中进行累积并定期写入磁盘。如果遇到文件结束标志，并且该文件将被关闭，则剩余的所有缓冲区内容都将写入磁盘，更新 i 节点的文件大小和块指针信息，并且对文件系统的可用磁盘块列表进行更新。要使文件系统完整性得到最大保护，将按照特定顺序对这些操作进行处理，如果在写入磁盘时发生系统崩溃或电源故障，则该特定顺序将尽可能减小磁盘上的文件系统损坏的风险。该顺序更新进程称为 “同步写入操作”。

HP-UX 文件系统在磁盘设备上随机广泛分布的位置中保存可用空间列表、块、i 节点以及其他文件部分。这就意味着在执行写入操作之前，以特定顺序写入文件信息块需要额外的时间来移动到磁盘上的所需位置。如果在此顺序中发生电源故障或系统崩溃，则将有一个或多个块无法正确更新，从而产生可能不一致的文件系统。 **fsck** 命令用于修复这种不一致性。

由于异步写入操作与 **fs_async** 内核参数有关，因此可允许系统以更方便（从而更快）的顺序而不是以更安全（更安全但是更慢）的顺序更新磁盘上的文件系统信息，从而减少了写入之间的搜索和移动延迟。然而，如果执行这

些操作时发生系统崩溃，则无法通过 **fsck** 自动修复的不一致文件系统的风险要显著地大于使用同步写入。

崩溃的结果

如果只使用同步写入操作，则对目录、文件 i 节点、可用空间列表等的所有更新都以 **fsck** 已知的顺序进行处理。如果顺序更新任意磁盘块时发生崩溃，则 **fsck** 可以很容易地确定崩溃发生的位置，并且会修复丢失的更新信息，很可能不需要系统管理员的帮助。

如果将 **fs_async** 设置为允许异步写入并且发生崩溃，则 **fsck** 将无法知道使用的顺序，因此在修复不一致文件系统信息、修复目录和 i 节点条目等时，可能需要管理员提供交互的帮助。

为何允许异步写入?

当对文件进行写操作之后关闭了文件时，等待同步写入操作和更新磁盘块将降低要求常用文件和目录进行写入和关闭操作的程序和应用程序的性能。允许异步写入操作显著地减少了这些延迟，从而在性能上产生了相应提高。但是，如果应用程序是使用磁盘 I/O 操作相对较少的 CPU 密集型应用程序时，则性能的提高非常少。

何时应使用异步写入?

为了提高系统性能，建议在下列情况使用异步写入操作:

- 电源故障的风险很低（非常依赖于电源和（或）不可间断电源）。
- 已采取预防措施来增强数据安全（复杂的文件系统备份或冗余策略），或者由于系统崩溃造成的可能数据丢失不如系统性能重要。
- 用户应用程序需要频繁打开、写入以及关闭磁盘文件和目录。
- 删除同步写入操作将有效提高系统性能来弥补任何相关风险。

要启用异步写入操作，请将 **fs_async** 内核参数设置为 **1** ，而不是缺省值 **0** 。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

fs_async 由 HP 开发。

另请参阅

fsck(1M)、 kctune(1M)、 sam(1M)、 gettune(2)、 open(2)、 settune(2)。

名称

fs_symlinks - 用于解析路径名的符号链接的最大数量

值

保证安全

20

缺省值

20

允许值

允许的最小值为 20。允许的最大值为 1024 或 1K。

指定一个正整数值。

说明

fs_symlinks 可调参数表示对路径名进行解析时，后接内核的符号链接的最大数量。创建符号链接和（或）解析路径名的应用程序，目前也应当使用这个由 **fs_symlinks** 表示的限制，以与内核保持一致。

更改此可调参数的人员

希望通过使用路径名来运行应用程序的任何用户，该路径名可能扩展为大量的符号链接。

更改限制

无。 **fs_symlinks** 可调参数是动态的（系统运行时，调整将立即生效）。

应该何时增加此可调参数的值

当应用程序应该创建和（或）解析可能扩展为大量符号链接的路径名时，应增加此可调参数的值。

增加此值的负面影响

无。

应该何时降低此可调参数的值

除非存在限制扩展路径名中符号链接数量的原因，否则很少需要降低此参数值。

降低此值的负面影响

如果符号链接数量超出了由 **fs_symlinks** 表示的限制，则内核将无法解析路径名。

应该同时更改的其他可调参数值

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

注释：传统上在 **<sys/param.h>** 中定义的 **MAXSYMLINKS** 目前已过时，不应使用。现在通过新的可调参数 **fs_symlinks** ，可在应用程序中使用此限制（在路径名中可能扩展的符号链接数量）。应当使用由可调参数基础结

构提供的接口，来获得 **fs_symlinks** 的值。在应用程序级别，请使用 *gettune*(2) 或 *kctune*(1M) 接口。使用 **MAXSYMLINKS** 定义的应用程序可能与内核不一致。在 HP-UX 11i v2 后来的版本中，将删除 **MAXSYMLINKS** 定义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

fs_symlinks 由 HP 开发。

另请参阅

kctune(1M)、 sam(1M)、 gettune(2)、 settune(2)、 ulimit(2)、 setrlimit(2)、 maxfiles_lim(5)。

名称

fs_wrapper - 文件系统管理命令使用的配置和二进制文件

概要

ff [**-F** *FStype*] ...

fsck [**-F** *FStype*] ...

fsdb [**-F** *FStype*] ...

labelit [**-F** *FStype*] ...

mkfs [**-F** *FStype*] ...

mount [**-F** *FStype*] ...

ncheck [**-F** *FStype*] ...

newfs [**-F** *FStype*] ...

quot [**-F** *FStype*] ...

quotacheck [**-F** *FStype*] ...

volcopy [**-F** *FStype*] ...

说明

"概要"中所列的命令可在不同类型的文件系统上运行。每个命令（ **mount** 除外）读取控制命令行为的文件系统特定配置文件，并且调用文件系统特定二进制文件进行实际操作。 *FStype* 是命令行中指定的文件系统类型。如果 *FStype* 未给定，则通过使用由命令提供的设备 **special** ，与 **/etc/fstab** 文件中的条目相匹配，来确定文件系统类型（有关命令用法的详细信息，请参阅单个命令）。

管理员也可以通过文件 **/etc/default/fs** ，为以上命令定义一个缺省的文件系统类型。如果此文件存在，并且包含下述行：

LOCAL=*FStype*

（例如， **LOCAL=hfs** ），则除非命令行或者 **/etc/fstab** 中提供了 *FStype* ，否则以上命令将采用在 **/etc/default/fs** 中给定的 *FStype* 。提供缺省文件系统规范，用于和 10.0 之前的命令调用保持兼容。

有关所用文件列表的详细信息，请参阅"文件"一节。

警告

配置文件 **/sbin/lib/mfsconfig.d/***FStype* 由 HP 或其他文件系统供应商提供。它们不是由系统管理员进行编辑。破坏或删除这些文件可能导致异常行为，包括无法引导。

配置文件的格式可以更改。

文件系统特定二进制文件通常不直接执行。但是，如果配置文件无法使用，则直接执行这些二进制文件对于修复和再次运行系统，可能是一个有用的步骤。二进制文件接受与执行这些文件的命令相同的参数。

mount 命令是一个特殊情况。该命令当前不读取配置文件，并且如果 *FStype* 为 **cdfs** 、 **hfs** 、 **nfs** 或 **lofs** ，该命令不执行文件系统特定的二进制文件。处理这些 *FStype* 的二进制文件也处理其他的 *FStype* ，同时还调用相应的文件系统特定的命令。

由于历史原因， **hfs** 二进制文件也可处理 **nfs** 和 **cdfs** ，因此没有单独的二进制文件用于后两个文件系统。

如果对这些命令（ **mount** 除外）进行重新命名，则它们将不能运行，因为它们是通过符号链接到单个的可执行文件 (**/sbin/fs_wrapper**) 的。

文件

FStype 是命令行中可以选择指定的文件系统类型。 *command* 是指命令的名称。

/sbin/fs/*FStype***/***command*	用于 **fsck** 、 **fsdb** 、 **mkfs** 、 **mount** 和 **newfs** 等命令的文件系统特定的二进制文件。在与 **fs_wrapper** 无关联的目录中，可能有其他的文件系统特定的二进制文件。
/usr/lbin/fs/*FStype***/***command*	用于其余命令的文件系统特定的二进制文件。在与 **fs_wrapper** 无关联的目录中，可能有其他的文件系统特定的二进制文件。
/sbin/lib/mfsconfig.d/*FStype*	每个文件系统类型的配置文件。
/etc/default/fs	可在其当中定义缺省文件系统类型的文件。如果此文件不存在，则没有缺省文件系统类型。
/etc/fstab	有关文件系统的静态信息

另请参阅

ff(1M) 、 fsck(1M) 、 fsdb(1M) 、 mkfs(1M) 、 mount(1M) 、 ncheck(1M) 、 newfs(1M) 、 quot(1M) 、 quotacheck(1M)、 volcopy(1M)、 fstab(4)。

名称

gssapi - 常规安全服务应用程序编程接口

说明

此简介包括在 RFC 2743“Generic Security Service Application Programming Interface”和 RFC 2744“Generic Security Service API:C-bindings”中定义的关于常规安全服务应用程序编程接口 (GSSAPI) 的常规信息，同时还包括对错误处理、数据类型和调用约定的概述，其中包括下列内容：

整型
字符串和其他类似数据类型
对象标识符 (OID)
对象标识符集（OID 集）
凭证
相关环境
验证标记
主要状态值
次要状态值
名称
通道绑定
可选参数

常规信息

常规安全服务应用程序编程接口 (GSSAPI) 可为使用对等通信的应用程序提供安全服务。通过使用 GSSAPI 例行程序，应用程序可执行下述操作：

使一个应用程序验证另一个应用程序的用户。

使一个应用程序将访问权限授予另一个应用程序。

按消息应用安全服务，如保密性和完整性。

GSSAPI 支持在两个正在通信的应用程序间建立安全连接。建立安全连接的应用程序称为 *context initiator*。接收安全连接的应用程序称为 *context acceptor*

在使用 GSSAPI 的过程中涉及四个阶段：

context initiator 获取可用于向其他进程证明其身份的凭证。同样，context acceptor 将获取可使其接收安全环境的凭证。任一应用程序都可省略此凭证获取过程，并在后续的各阶段中使用其缺省凭证。有关详细信息，请参阅本联机帮助页的“凭证”一节。

这些应用程序使用凭证来建立其全局标识。全局标识可以（但不一定）与该应用程序运行时所使用的本地用户名相关。凭证可包含下面几项中的任一项：

登录环境　　登录环境包括主体的网络凭证，以及其他帐户信息。

安全环境　　　　　　正在通信的应用程序通过交换验证标记来建立共同的安全环境。

安全环境是一对 GSSAPI 数据结构，其中包含在通信的应用程序间共享的信息。该信息描述每一个应用程序的状态。此安全环境对于按消息应用的安全服务来说是必需的。

要建立一个安全环境， context initiator 将调用 **gss_init_sec_context()** 例行程序来获取一个 标记。该标记是受口令保护的不透明数据。 context initiator 将该标记传输给 context acceptor，而 context acceptor 又将该标记传递给 **gss_accept_sec_context()** 例行程序，以便对共享信息进行解码和提取。

作为建立安全环境的一部分， context acceptor 将对 context initiator 进行验证，而 context initiator 可以反过来要求 context acceptor 对其自身进行验证。

context initiator 可授权给 context acceptor，使其充当其代理。授权意味着 context initiator 将 context acceptor 用作代理，并授予它启动更多安全环境的权限。要授权，context initiator 应在 **gss_init_sec_context()** 例行程序上设置一个标记，表示其想要以常规方式向 context acceptor 授权并发送返回的标记。此 acceptor 将此标记传递给 **gss_accept_sec_context()** 例行程序，从而生成授权凭证。 context acceptor 可使用该凭证启动更多安全环境。这些应用程序可使用此环境来交换受保护的消息和数据。

应用程序可调用 GSSAPI 例行程序以保护消息中交换的数据。应用程序通过调用相应的 GSSAPI 例行程序来发送一条受保护的消息，以便执行下述操作：

　　应用保护。

　　将消息绑定到相应的安全环境中。

然后该应用程序可将得到的信息发送给对等端应用程序。

接收该消息的应用程序将收到的数据传递给一个 GSSAPI 例行程序，该例行程序将去除保护并验证数据的有效性。

GSSAPI 将应用程序数据作为任意八进制字符串进行处理。按消息应用的 GSSAPI 安全服务可实现下列目的之一：

　　数据来源的完整性以及正确验证。

　　数据来源的保密性、完整性以及正确验证。

当应用程序完成通信后，其中任意一项都可指示 GSSAPI 删除该安全环境。

标准的 GSSAPI 例行程序在“Internet RFC 2743， Generic Security Service Application Programming Interface”和“RFC 2744，Generic Security Service API:C-bindings”中进行了定义。这些例行程序具有前缀 **gss_** 。

下列各节将概述 GSSAPI 错误处理和数据类型。

错误处理

每个 GSSAPI 例行程序返回主次两个状态值:

主要状态值　　主要状态值是在 RFC 2744 中定义的常规 API 例行程序错误或调用错误。

次要状态值　　次要状态值指示此机制专有的错误。

如果一个例行程序的输出参数中包含该例行程序所分配的存储空间的指针，则即使该例行程序返回错误，这些输出参数仍会始终包含一个有效的指针。如果未分配存储空间，则该例行程序会将该指针设置为 NULL，并将与这些指针相关联的任意 length 字段（如 **gss_buffer_desc** 结构中的字段）设置为 0（零）。

次要状态值通常包含有关错误的更多详细信息。然而，这些值不能在各个 GSSAPI 实现方案之间进行移植。当编写可移植的应用程序时，请使用主要状态值来处理错误。可使用次要状态值来调试应用程序，并向用户显示错误和错误恢复信息。

GSSAPI 数据类型

本节将概述 GSSAPI 数据类型及其定义。

整型　　GSSAPI 可定义下列整型数据类型:

OM_uint32　**32** 位无符号整数

此整型数据类型是一种可移植的数据类型，GSSAPI 例行程序定义使用这种数据类型来保证最低位数。

字符串和类似数据类型

许多 GSSAPI 例行程序可采用参数，并返回用来描述连续多字节数据的值，例如，不透明数据和字符串。 **gss_buffer_t** 数据类型是一个指向缓冲区描述符 **gss_buffer_desc** 的指针，使用它可在 GSSAPI 例行程序和应用程序之间传递数据。

gss_buffer_t 数据类型具有下述结构:

```
typedef struct gss_buffer_desc_struct {
      size_t  length;
      void    *value;
} gss_buffer_desc, *gss_buffer_t;
```

length 字段包含数据的总字节数，而 value 字段中包含一个指向实际数据的指针。

当使用 **gss_buffer_t** 数据类型时，GSSAPI 例行程序将为传递给应用程序的任何数据分配存储空间。调用应用程序必须分配 **gss_buffer_desc** 对象。它可使用值 **GSS_C_EMPTY_BUFFER** 来初始化尚未使用的 **gss_buffer_desc** 对象。要释放存储空间，应用程序应调用 **gss_release_buffer()** 例行程序。

对象标识符　　应用程序使用 **gss_OID** 数据类型来选择安全机制（例如，Kerberos），并指定名称类型。可使用相应的 OID 来选择安全机制:

对于 Kerberos v5，请指定 **GSS_C_OID_KRBV5_DES** 。

GSS_C_NULL_OID 有助于确保应用程序的可移植性。

gss_OID 数据类型包含由 ISO 定义的树形结构值，并具有下述结构：

```
typedef struct gss_OID_desc_struct {
        OM_uint32 length;
        void    * elements;
  }  gss_OID_desc, *gss_OID;
```

该结构的 elements 字段指向一个八进制字符串的第一个字节，该字符串包含 **gss_OID** 数据类型的值的 ASN.1 BER 编码。 length 字段包含该值中的字节数。

从 GSSAPI 返回的 **gss_OID_desc** 值是只读的。应用程序不应尝试取消分配这些值。

对象标识符集 **gss_OID_set** 数据类型表示一个或多个对象标识符。 **gss_OID_set** 数据类型的值用于：

报告 GSSAPI 所支持的可用机制。

请求特定的机制。

指示凭证所支持的机制。

gss_OID_set 数据类型具有下述结构：

```
  typedef struct gss_OID_set_desc_struct {
        int     count
        gss_OID elements
  } gss_OID_set_desc, *gss_OID_set;
```

count 字段包含集合中 OID 的数量。 elements 字段是指向 **gss_oid_desc** 对象数组的指针，并且数组中的每个对象都描述一个 OID。应用程序调用 **gss_release_oid_set()** 例行程序来取消分配与 GSSAPI 例行程序返回给该应用程序的 **gss_OID_set** 值相关联的存储空间。

凭证

凭证用于建立或检验应用程序或其他主体的身份

gss_cred_id_t 数据类型是一种基本数据类型，用于标识 GSSAPI 凭证数据结构。

相关环境

安全环境是一对 GSSAPI 数据结构，其中包含正在通信的应用程序间共享的信息。该信息描述每个应用程序的加密状态。此安全环境是按消息应用的安全服务所必需的，并通过成功进行验证交换来创建。

gss_ctx_id_t 数据类型包含一个用于标识 GSSAPI 安全环境其中一端的基本值。该数据类型对于调用方来说是不透明的。

验证标记

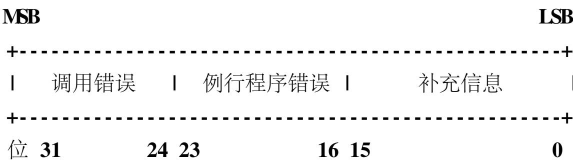
GSSAPI 使用标记来使共享一个安全环境的两个应用程序之间保持同步。该标记是受口令保护的位字符串，在 GSSAPI 安全环境其中一端由安全机制生成，供安全环境另一端的对等端应用程序使用。该数据类型对于调用方来说是不透明的。

这些应用程序将 **gss_buffer_t** 数据类型用作 GSSAPI 例行程序的标记。

主要状态值

GSSAPI 例行程序返回 GSS 状态代码作为其 **OM_uint32** 函数值。这些代码表示常规 API 例行程序错误或调用错误。

一个 GSS 状态代码可以表示一个来自例行程序的常规致命 API 错误和一个调用错误。 GSS 状态代码中还可包含其他状态信息。这些错误将编码为 32 位 GSS 状态代码，如下所示：



如果 GSSAPI 例行程序返回一个 GSS 状态代码，该代码的第 16 位包含一个非零值，则意味着调用失败。如果“调用错误”字段不为零，则意味着 context initiator 使用例行程序时出错。另外，例行程序可通过设置状态代码的补充信息字段中的各个位来指示附加信息。下文的表介绍了例行程序错误、调用错误，以及补充信息状态位及其含义。

下表列出了各个 GSSAPI 例行程序错误及其含义：

GSSAPI 例行程序错误

名称	字段值	含义
GSS_S_BAD_MECH	1	所需的机制不受支持。
GSS_S_NAME	2	传递的名称无效。
GSS_S_NAMETYPE	3	传递的名称不受支持。
GSS_S_BAD_BINDINGS	4	通道绑定不正确。
GSS_S_BAD_STATUS	5	状态值无效。
GSS_S_BAD_SIG	6	标记包含无效的签名。
GSS_S_NO_CRED	7	未提供凭证。
GSS_S_NO_CONTEXT	8	尚未建立环境。
GSS_S_DEFECTIVE_TOKEN	9	标记无效。
GSS_S_DEFECTIVE_CREDENTIAL	10	凭证无效。
GSS_S_CREDENTIALS_EXPIRED	11	引用的凭证已过期。
GSS_S_CONTEXT_EXPIRED	12	环境已过期。

GSS_S_FAILURE	13	例行程序失败。检查次要状态代码。
GSS_S_BAD_QOP	14	无法提供请求的保护质量。
GSS_S_UNAUTHORIZED	15	本地安全策略禁止该操作。

下表列出了各调用错误值及其含义:

调用错误

名称	字段值	含义
GSS_S_CALL_INACCESSIBLE_READ	1	无法读取所需的输入参数。
GSS_S_CALL_INACCESSIBLE_WRITE	2	无法写入所需的输出参数。
GSS_S_BAD_STRUCTURE	3	参数未正确构建。

下表列出了各补充位及其含义。

补充信息状态位

名称	位编号	含义
GSS_S_CONTINUE_NEEDED	0 (LSB)	再次调用该例行程序以完成其功能。
GSS_S_DUPLICATE_TOKEN	1	该标记是较早标记的副本。
GSS_S_OLD_TOKEN	2	该标记的有效期已过期；例行程序无法验证该标记不是较早标记的副本。
GSS_S_UNSEQ_TOKEN	3	已处理一个后续标记。

所有的 **GSS_S_** 符号等同于完整的 **OM_uint32** 状态代码，而不是位字段值。例如， **GSS_S_BAD_NAMETYPE** 的实际值（“例行程序错误”字段中的 3）为 **3 << 16** 。

主要状态代码 **GSS_S_FAILURE** 表示基础安全机制已检测到一个错误，并且没有可用于该错误的主要状态代码。有关该错误的详细信息，请检查次要状态代码。有关详细信息，请参阅“次要状态值”一节。

GSSAPI 提供三个宏:

GSS_CALLING_ERROR()

GSS_ROUTINE_ERROR()

GSS_SUPPLEMENTARY_INFO()

每一个宏采用一个 GSS 状态代码，并屏蔽除相关字段之外的所有字段。例如，当在一个状态代码上使用 **GSS_ROUTINE_ERROR()** 宏时，该宏返回一个值。可以通过仅使用“例行程序错误”字段，并将“调用错误”和“补充信息”字段置零，可得到该宏的值。

附加宏 **GSS_ERROR()** 允许用户确定该状态代码是表示调用错误还是例行程序错误。如果状态代码指示一个调用或例行程序错误，则宏返回一个非零值。如果不表示调用或例行程序错误，则宏返回 0（零）。

注释：有时，无法访问数据的 GSSAPI 例行程序可生成一个特定于平台的信号，而不是返回 **GSS_S_CALL_INACCESSIBLE_READ** 或 **GSS_S_CALL_INACCESSIBLE_WRITE** 状态值。

次要状态值

GSSAPI 例行程序返回 *minor_status* 参数，表示来自基础安全机制的错误。该参数可包含由 **OM_uint32** 数据类型值所表示的单个错误。

名称

名称可用于标识主体。 GSSAPI 可验证名称与声明该名称的主体之间的关系。

名称有两种表示形式：

可打印形式，用于显示应用程序。

内部规范形式，供 API 使用，对于应用程序来说是不透明的。

gss_import_name() 和 **gss_display_name()** 例行程序可在可打印形式和 **gss_name_t** 数据类型之间对名称进行转换。

gss_compare_name() 例行程序用于比较内部形式的名称。

通道绑定

用户可定义和使用通道绑定，以便使安全环境与传送该环境的通信通道相关联。通过使用下列结构，可将通道绑定传送给 GSSAPI：

```
typedef struct gss_channel_binding_struct {
    OM_uint32       initiator_addrtype;
    gss_buffer_desc initiator_address;
    OM_uint32       acceptor_addrtype;
    gss_buffer_desc aceptor_address;
    gss_buffer_desc application_data;
} *gss_channel_bindings_t;
```

使用 **initiator_addrtype** 和 **acceptor_addrtype** 字段启动在 **initiator_address** 和 **acceptor_address** 缓冲区中包含的地址的类型。下表列出了各地址类型及其 **addrtype** 值：

地址类型	**addrtype** 值
未指定	GSS_C_AF_UNSPEC
Host-local	GSS_C_AF_LOCAL
DARPA Internet	GSS_C_AF_INET
ARPAnet IMP	GSS_C_AF_IMPLINK
pup 协议（例如，BSP）	GSS_C_AF_PUP

MIT CHAOS 协议	GSS_C_AF_CHAOS
XEROX NS	GSS_C_AF_NS
nbs	GSS_C_AF_NBS
ECMA	GSS_C_AF_ECMA
datakit 协议	GSS_C_AF_DATAKIT
CCITT 协议（例如，X.25）	GSS_C_AF_CCITT
IBM SNA	GSS_C_AF_SNA
Digital DECnet	GSS_C_AF_DECnet
直接数据链接接口	GSS_C_AF_DLI
LAT	GSS_C_AF_LAT
NSC 超信道	GSS_C_AF_HYLINK
AppleTalk	GSS_C_AF_APPLETALK
BISYNC 2780/3780	GSS_C_AF_BSC
分布式系统服务	GSS_C_AF_DSS
OSI TP4	GSS_C_AF_OSI
X25	GSS_C_AF_X25
未指定地址	GSS_C_AF_NULLADDR

这些标记指定地址系列，而不是地址格式。对于包含多种备用地址形式的地址系列， **initiator_address** 和 **acceptor_address** 字段应包含充足的信息来确定所使用的地址形式。按照字节在网络上传递的顺序设置包含地址的字节的格式。

GSSAPI 通过连接所有的字段（ **initiator_addrtype** 、 **initiator_address** 、 **acceptor_addrtype**、 **acceptor_address** 和 **application_data** ）来创建一个八进制字符串。安全机制对该八进制字符串进行签名，并将该签名绑定到由 **gss_init_sec_context()** 例行程序生成的标记上。 context acceptor 向 **gss_accept_sec_context()** 例行程序提供相同的绑定，该例行程序将对签名进行评估，并将其与标记中的签名进行比较。如果这两个签名不同，则 **gss_accept_sec_context()** 例行程序返回 **GSS_S_BAD_BINDINGS** 错误，并且不建立该环境。

一些安全机制将检查提供给 **gss_init_sec_context()** 例行程序的通道绑定的 **initiator_address**

字段是否包含主机系统的正确网络地址。因此，可移植的应用程序应对 **initiator_addrtype** 地址字段使用正确的地址类型和值或 **GSS_C_AF_NULLADDR** 。一些安全机制包含通道绑定数据而不是签名中的标记，因此，可移植的应用程序不应将机密数据用作通道绑定的组成部分。 GSSAPI 不验证该地址，也不包括标记中的纯文本绑定信息。

可选的参数

在例行程序描述中，通过可选参数，可以使应用程序通过传递缺省值来请求缺省行为。下列约定适用于可选参数：

约定	缺省值	解释

gss_buffer_t 类型	GSS_C_NO_BUFFER	对于输入参数，表示未提供任何数据。对于输出参数，表示返回的信息不是应用程序所必需的。
整型（输入）		有关缺省值，请参阅参考页。
整型（输出）	NULL	表示应用程序不需要该信息。
指针类型（输出）	NULL	表示应用程序不需要该信息。
OID	GSS_C_NULL_OID	表示名称类型或安全机制的缺省选择。
OID 集	GSS_C_NULL_OID_SET	表示安全机制的缺省集合
凭证	GSS_C_NO_CREDENTIAL	表示该应用程序应使用缺省的凭证句柄。
通道绑定	GSS_C_NO_CHANNEL_BINDINGS	表示未使用任何通道绑定。

另请参阅

gss_accept_sec_context(3)、 gss_compare_name(3)、 gss_display_name(3)、 gss_import_name(3)、 gss_init_sec_context(3)、 gss_release_buffer(3)、 gss_release_oid_set(3)、 libgss(4)。

RFC 2743、RFC 2744。

DCE-GSSAPI 联机帮助页随 DCE-CoreTools 产品一同提供。要查看这些联机帮助页，请将 **/opt/dce/share/man** 添加到 **MANPATH** 中。

名称

gvid_no_claim_dev - gvid 图形驱动程序不支持的 PCI 供应商（或设备）ID

值

保证安全

0

缺省值

0

允许值

0 至 **0xFFFFFFFF**（**4294967295**）

建议值

针对不希望 **gvid** 支持的设备。

说明

此可调参数指定了 HP **gvid** 图形驱动程序不应支持的图形设备的 PCI 供应商 ID 和（或）设备 ID。

从设计上而言， **gvid** 图形驱动程序是一个适用于所有 PCI 图形卡的通用驱动程序。因此， **gvid** 驱动程序将尝试支持任何能在系统上找到的 PCI 图形显示设备。对于需要使用自己的图形驱动程序支持自己的图形设备的图形设备开发商，这可能会是一个问题。

gvid_no_claim_dev 的值将指定 gvid 图形驱动程序不应支持的图形卡的 PCI 供应商 ID 和（或）设备 ID。它是一个 32 位数，高 16 位包含 PCI 供应商 ID，低 16 位包含 PCI 设备 ID。供应商 ID 或设备 ID 都可使用形式为 **0xFFFF** 的通配符值。

例如，如果将 **gvid_no_claim_dev** 设置为值 **0x1234ABCD** ，那么 **gvid** 驱动程序将忽略 PCI 供应商 ID 为 **0x1234**、设备 ID 为 **0xABCD** 的图形设备。如果将 **gvid_no_claim_dev** 设置为 **0x4567FFFF** ，那么无论其设备 ID 如何， **gvid** 驱动程序将不会支持其供应商 ID 为 **0x4567** 的任何图形卡。

更改此可调参数的人员

希望 **gvid** 图形驱动程序忽略特定图形设备的任何人。这通常将是那些为自己的硬件编写自己的图形驱动程序的人。

更改限制

对此可调参数的更改将在下次重新引导时生效。

应该同时更改的其他可调参数

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在

用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

gvid_no_claim_dev 由 HP 开发。

名称

hdlpreg_hash_locks - 确定 pregion spinlock 池的大小

说明

此可调参数在以前版本的 HP-UX 中使用，以确定在同步 pregion 子结构的变更中分配要使用的 spinlock 的数量。自从 HP-UX 11i v3.0 和更高版本，子结构不再使用这种方式同步，池也不再存在。因此，将不再需要此可调参数，此可调参数从 HP-UX 11i v3.0 起已过时。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

hdlpreg_hash_locks 由 HP 开发。

名称

hfs_revra_per_disk - 当向后顺序读取时，通过一个预读操作读取的 HFS 文件系统块的最大数量

值

保证安全

64

缺省值

64

允许值

0 到 **8192**

说明

此可调参数定义了向后顺序读取时，通过一个预读操作读取的 HFS 文件系统块的最大数量。

更改此可调参数的人员

由 HP 现场服务工程师更改，而不是直接由客户更改。

应该何时增加此可调参数的值

如果在具有小文件系统块的文件系统中，有大量的反向顺序文件 I/O，则应增加此可调参数的值。

增加此值的负面影响

系统将消耗更多缓冲区缓存中的内存。

应该何时降低此可调参数的值

如果在具有大文件系统块大小的文件系统中，有很少量的反向顺序文件 I/O，则应降低此可调参数的值。

降低此值的负面影响

文件处理速度将降低。

应该同时更改的其他可调参数

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

hfs_revra_per_disk 由 HP 开发。

名称

hier - 文件系统层次结构

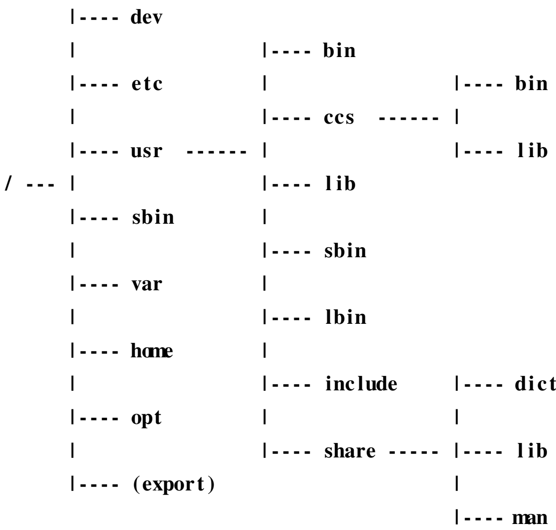
说明

HP-UX 文件系统是为了便于管理而组织成树状层次结构。在文件系统树状结构中，将为一台计算机的专用文件、可以被多台计算机共享的文件以及主目录提供不同的区域。

有两种类型的共享文件：可以由相同体系结构的多台计算机共享的文件，以及可以由所有计算机共享的文件。通过这种组织形式，可共享的文件可以保存在一台计算机（服务器）上，但是由很多计算机（客户端）访问。

下图说明了文件系统的布局。请注意，有很多目录没有在此图中出现，但是在下面进行了讨论。

“目录布局图”



下面的列表讨论了典型的 HP-UX 目录层次结构。某些 HP-UX 应用程序可能会添加额外的未显示的目录。

/ 根目录。

/dev 专用文件（块和字符设备文件）；请参阅 *mknod*(1M)。

/etc 特定主机的配置和管理数据库。

/etc/opt 特定应用程序配置文件的目录（可选程序包的配置信息）。

/etc/rc.config.d 启动配置文件。

/export 导出文件系统的缺省根目录。仅适用于服务器。

/home 用户目录的缺省根目录。

/lost+found 连接分离文件的保存目录；供 *fsck*(1M) 使用。

/mnt 本地文件系统的挂接点。

/net 远程文件系统的挂接点。

/opt 可选应用程序包的子树的根目录。

/sbin 基本系统命令。基本命令定义为引导系统和挂接文件系统所需的可执行命令。只有在挂接 **/usr** 之后，才可获得实用程序的完整补充。

/sbin/init.d 启动和关闭脚本。

/sbin/rc0.d 要进入或离开运行级别 0，请将文件链接到 **/sbin/init.d** 中的脚本。

/sbin/rc1.d 要进入或离开运行级别 1，请将文件链接到 **/sbin/init.d** 中的脚本。

/sbin/rc2.d 要进入或离开运行级别 2，请将文件链接到 **/sbin/init.d** 中的脚本。

/sbin/rc3.d 要进入或离开运行级别 3，请将文件链接到 **/sbin/init.d** 中的脚本。

/stand 独立二进制代码和内核配置文件。

/tmp 系统生成的临时文件；通常在引导操作过程中清除。

/usr 可共享的用户和系统管理命令、库和文档的挂接点。

/usr/bin 常用实用程序和用户命令的主要位置。

/usr/ccs C 编译系统。用于生成 C 程序的工具和库。

/usr/ccs/bin 开发二进制代码；包括 **cc**、**make**、**strings** 等。

/usr/ccs/lib 开发库。

/usr/ccs/lbin 开发后端。

/usr/conf 内核配置文件。

/usr/contrib 用户提供的（不支持的、内部的）命令、文件等的目录。在此目录中的文件来自于本地站点或者组织之外（例如，来自用户组或 HP 服务工程师）。

/usr/contrib/bin 用户提供的命令。

/usr/contrib/include 用户提供的包含文件。

/usr/contrib/lib 用户提供的库。

/usr/contrib/man 用户提供的联机帮助页。

/usr/include 包括 C 和其他程序的头文件。下面列出了部分子目录。

/usr/include/machine	特定计算机的 C 包含文件。
/usr/include/nfs	网络文件系统 (NFS) 的 C 包含文件。
/usr/include/sys	内核相关的 C 语言头文件。
/usr/lbin	其他命令后端可执行文件的目录。后端可执行文件是通常不由用户直接调用的可执行程序。
/usr/lib	程序库、对象代码和基于体系结构的数据库。
/usr/lib/nls	本地语言支持的目录。
/usr/local	站点本地目录、文件等的目录。在此目录下的文件来自本地站点或组织内部。有关非本地的不受支持的命令和文件，请参阅 **/usr/contrib**。
/usr/local/bin	站点本地命令。
/usr/local/lib	站点本地库。
/usr/local/man	站点本地联机帮助页。
/usr/newconfig	缺省操作系统配置数据文件。此目录是目录层次结构镜像 **/** 。将可以自定义的配置文件和数据库的新版本放置在此，从而不会覆盖当前的版本。对新安装的系统，将此目录中的文件复制到常规位置。系统管理员可能会希望保存它们供以后参考。
/usr/old	已停用或过时的文件和程序。
/usr/sbin	系统管理命令。
/usr/share	独立于体系结构的可共享文件。
/usr/share/dict	**spell** 和 **ispell** 的词典。
/usr/share/lib	其他可共享的库。
/usr/share/man	联机文档。
/var	“变动”文件子树的根目录。这些是运行时创建的文件，并且可以增长到任意大小。某些示例包括日志文件、临时文件、暂态文件和打印缓存文件。
/var/adm	系统管理文件，例如日志文件和记账文件。有的子目录列出如下。
/var/adm/crash	用于保存内核崩溃转储。
/var/adm/cron	*cron*(1M) 队列的目录。
/var/adm/sw	软件分发仓库的缺省位置。
/var/adm/syslog	**syslog** 生成的日志文件。请参阅 *syslog*(3C) 和 *syslogd*(1M) 。
/var/mail	收到的邮件。

/var/news *news*(1) 的本地系统新闻文章。

/var/opt 与可选软件程序包相关的变动文件子树的根目录。

/var/preserve *ex*(1) 和 *vi*(1) 保存丢失的编辑会话直到将其恢复为止的位置。

/var/run 守护程序运行时创建的文件。例如，放置在此的 **syslogd**、**syslog.pid** 的进程 ID (PID) 文件。

/var/spool 用于打印机打印缓存、传递邮件、 *cron*(1M) 等的其他目录。

/var/spool/cron *cron*(1M) 和 *at*(1) 打印缓存文件。

/var/spool/lp 打印机打印缓存文件。

/var/spool/mqueue 包含来自邮件系统的外发邮件和日志文件。

/var/spool/uucp UUCP 打印缓存目录。

/var/tmp 应用程序生成的临时文件。在系统重新引导时通常不清除此目录。

/var/uucp UUCP 管理文件。

相关内容

某些目录包括所有 HP-UX 实现不支持的命令或文件。

另请参阅

find(1)、 grep(1)、 ls(1)、 whereis(1)。

名称

hires_timeout_enable - 启用高精度计时器支持

值

保证安全

0

缺省值

0

允许值

0：禁用

1：启用

建议值

除非应用程序需要高精度计时器，否则请使用 **0** 。

说明

hires_timeout_enable 是一个动态可调参数，可启用（或禁用）对下列应用程序编程接口 (API) 的高精度计时器和定时休眠的支持 - **getitimer()** 、 **nanosleep()** 、 **semtimedop()** 、 **setitimer()** 、 **sigtimedwait()** 、 **timer_gettime()**、 **timer_settime()** 、 **ualarm()** 、 **usleep()** 和 **pthread_cond_timedwait()** 。如果使用更高的精度，这些接口应该可以支持比当前 10 毫秒更短的时间间隔。

更改此可调参数的人员

需要对应用程序使用高于缺省精度的系统管理员。

更改限制

hires_timeout_enable 更改立即生效。但是，更改此可调参数不影响已经排队的定时事件。

应该何时增加此可调参数的值

如果应用程序需要更高的精度进行正确的操作，则应将 **hires_timeout_enable** 的值设置为 **1** 。更改将是系统级的。

增加此值的负面影响

可能会增加计时器相关中断的数目。

应该何时降低此可调参数的值

如果不再需要将缺省精度覆盖为更高的精度，则可以将 **hires_timeout_enable** 的值设置为 **0** 。

降低此值的负面影响

计时器相关的中断更少。

应该同时更改的其他可调参数值

无

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

hires_timeout_enable 由 HP 开发。

另请参阅

getitimer(2)、 nanosleep(2)、 semop(2)、 setitimer(2)、 sigtimedwait(2)、 timer_gettime(2)、 timer_settime(2)、 ualarm(2)、 usleep(2)、 pthread_cond_wait(3T)。

名称

hostname - 主机名解析说明

说明

主机名即为域。一个域是由多个子域组成的具有层次结构的、以点分隔的列表。例如，Internet 域名系统 (DNS) 中 **EDU** 子域的 **Berkeley** 子域中的计算机 **monet** ，可表示为

monet.Berkeley.EDU

（无结尾点）。

主机名经常用于网络客户端和服务器程序，通常必须将名称转换成地址以便使用（这个任务通常由库例行程序 **gethostbyname()** 来执行）。

当使用 NIS 或主机表解析主机名时，将在无需修改的情况下查找主机名。如果使用 DNS，则解析程序可将域添加到主机名。

通过 Internet 名称解析程序解析主机名的缺省方法遵循 **RFC 1535** 的安全建议。管理员可以采取操作来覆盖这些建议，并使解析程序以与早期不遵循 RFC 1535 的解析程序相同的方式运行。

缺省方法（使用 RFC 1535 标准）如下:

如果名称由单一部分组成，即不含点，并且环境变量 **HOSTALIASES** 设置为一个文件名，则将搜索该文件以查找与输入的主机名匹配的字符串。该文件应包含以空格分隔的两个字符串组成的行，第一个字符串为主机名别名，第二个字符串为用于替换别名的完整主机名。如果在要解析的主机名和文件行的第一个字段中，发现一个不区分大小写的匹配项，则将查找替换的名称而不会进一步处理。

如果名称中至少有一个点，则首先将其视为名称。引起此操作的点的数量是可配置的，方法是使用 **/etc/resolv.conf** 中的 **ndots** 选项来设置阙值（缺省为： **1** ）。如果名称以一个点结尾，则删除结尾点，只查找名称的其余部分（不考虑“ndots”选项的设置），并且不必再作进一步处理。

如果输入的名称不是以结尾点结尾，则将通过搜索域列表查找该名称，直到发现一个匹配项为止。如果 **/etc/resolv.conf** 文件中的搜索选项或 **LOCALDOMAIN** 环境变量都未使用，则域搜索列表只包括由域选项（在 **/etc/resolv.conf**
中）指定的全部域或者用于本地主机名中的域（请参阅 *resolver*(4) ）。例如，如果 **domain** 选项设置为 *CS.Berkeley.EDU* ，则搜索列表中将只有 CS.Berkeley.EDU，并且它是唯一追加到部分主机名 *lithium* 的域，使得 **lithium.BS.Berkeley.EDU** 成为使用搜索列表尝试搜索的唯一名称。

如果 **/etc/resolv.conf** 中使用了搜索选项，或者用户设置了环境变量 **LOCALDOMAIN** ，则搜索列表将包括由这些方法设置的内容。例如，如果 **search** 选项包含

CS.Berkeley.EDU CChem.Berkeley.EDU Berkeley.EDU

则将尝试为部分主机名（例如， *lithium* ）添加每一个域名（按照指定的相同顺序）。得到的可用于尝试的主机名如下:

lithium.BS.Berkeley.EDU
lithium.Bhem.Berkeley.EDU
lithium.Berkeley.EDU

环境变量 **LOCALDOMAIN** 覆盖了 **search** 和 **domain** 选项，并且如果两个选项都存在于解析程序配置文件中，则只使用后一个列出的选项（请参阅 *resolver*(4)）。

如果名称先前未尝试“作为名称”（即，降低至 **ndots** 阈值以下或者不含点），则将按照最初提供的名称进行尝试。

作者

hostname 由加州大学伯克利分校开发。

另请参阅

named(1M)、 gethostbyname(3N)、 gethostent(3N)、 resolver(4)、 RFC 1535。

名称

hosts_access - 主机访问控制文件的格式

说明

Internet 服务的访问控制功能使用访问控制文件来允许或拒绝对其服务的访问。这些文件是使用一种简单的访问控制语言定义的，其基于客户端（主机名/地址，用户名）和服务器（进程名，主机名/地址）模式。有关其快速简介，请参阅“举例”一节。

hosts_options(5) 中描述了该访问控制语言的扩展版本。

访问控制文件

daemon 是网络守护程序进程的进程名， **client** 是请求服务的主机名和（或）地址。网络守护程序进程名在 **inetd** 配置文件 (**/etc/inetd.conf**) 中指定。访问控制软件将搜索以下两个文件的内容： **/etc/hosts.allow** 和 **/etc/hosts.deny**。

搜索文件的顺序如下。找到第一个匹配后搜索即停止：

- 首先检查 **/etc/hosts.allow** 文件中的匹配对（守护程序，客户端）。如果找到一个匹配对，则授予访问权限并停止搜索。
- 如果未在 **/etc/hosts.allow** 文件中找到匹配对，则检查 **/etc/hosts.deny** 文件；如果在后者中找到一个匹配对（守护程序，客户端），则相应的访问将被拒绝。
- 如果在这两个访问控制文件中都未找到匹配对（守护程序，客户端），则授予访问权限。

如果不存在访问控制文件，系统会将其视为空文件处理。因此，我们可以通过不提供任何访问控制文件来关闭访问控制功能。

访问控制规则

每个访问控制文件都包含多行文本，也可能不包含任何文本。这些文本行按其出现的先后顺序被处理。当找到一个匹配后，即终止搜索。下面各项说明了访问控制文件的格式：

- 以反斜线“\”开头时，换行符将被忽略。可以借此断开较长的行以便于对其进行编辑。
- 空白行以及以 **#** 字符开头的行将被忽略。可以借此插入注释和空格，以使表格更易于阅读。
- 所有其他行均应采用以下格式。方括号 **[]** 中的内容是可选的：

  daemon_list **:** *client_list* [**:** *shell_command*]

  daemon_list 是一个或多个守护程序进程名 (**argv[0]** *values*) 或通配符（请参阅下面的内容）的列表。

client_list 是要与客户端主机名或地址进行匹配的一个或多个主机名、主机地址、模式或通配符（请参阅下面的内容）的列表。注释：IPv6 地址应包含在方括号 **[]** 中并且不能包含任何空格。

更为复杂的形式 **daemon@***host* 和 **user@***host* 分别在“服务器端点模式”和“客户端用户名查找”一节中进行了说明。

列表中的各元素必须用空格和（或）逗号分隔。

所有访问控制检查均区分大小写，但 NIS (YP) 网络组查找除外。

模式

访问控制语言采用以下模式：

- 以点 (**.**) 开头的字符串指定了要匹配点 后面的内容。如果某个主机名的最后几部分与指定的模式匹配，则该主机名就是匹配的。例如，模式 **.xyz.com** 与主机名 **abc.def.xyz.com** 就是匹配的。
- 以点 (**.**) 结束的字符串指定了要匹配点 前面的内容。如果某个主机地址的前几个数字字段与给定的字符串匹配，则该主机地址就是匹配的。例如，模式 **192.3.** 与 **192.3** 网络上的（几乎）每个主机 (192.3.x.x) 都是匹配的。
- 以 @ 字符开头的字符串被视为 NIS （以前的 YP）网络组名称。如果某个主机名是指定网络组的主机成员，则该主机名就是匹配的。守护程序进程名或客户端用户名不支持网络组匹配。
- *n.n.n.n***/***m.m.m.m* 形式的表达式被解释为一个“网络/掩码”对。如果“网络”与地址和“掩码”的按位“与”运算相等，则该主机地址就是匹配的。例如，网络/掩码模式 **131.155.72.0/255.255.254.0** 将与范围 **131.155.72.0** 到 **131.155.73.255** 中的每个地址相匹配。
- **[***IPv6_address***/***prefix_length***]** 形式的表达式被解释为一个 IPv6 网络前缀。如果 *IPv6_address* 中的 *prefix_length* 位的值与主机地址相等，则该主机地址就是匹配的。例如，模式 **[3ffe::1111:1234/120]** 将与范围 **3ffe::1111:0** 到 **3ffe::1111:ffff** 中的每个地址相匹配。

通配符

访问控制语言支持显式通配符。它们是：

ALL 通用通配符，始终匹配。

LOCAL 匹配任何名称中不包含点字符的主机。

UNKNOWN 匹配任何名称未知的用户，匹配任何名称或地址未知的主机。使用此模式时需小心；主机名可能会由于临时名称服务器的问题而不可用。如果软件无法确定它要与之对话的网络的类型，则网络地址将不可用。

KNOWN 匹配任何名称已知的用户，匹配任何名称和地址已知的主机。使用此模式时需小心；主机名可能会由于临时名称服务器的问题而不可用。如果软件无法确定它要与之对话的网络的类型，则网络地址将不可用。

PARANOID 匹配任何名称不与其地址相匹配的主机。如果 **/etc/tcpd.conf** 中的配置参数 *on_reverselookup_fail* 被设置为 **deny**，则 **tcpd** 会在查询访问控制表之前即拒绝来自这类客户端的请求。

运算符

访问控制语言支持以下运算符：

EXCEPT 此运算符的使用格式如下：

list_1 **EXCEPT** *list_2*

此结构将匹配任何与 *list_1* 匹配但不与 *list_2* 匹配的项。 **EXCEPT** 运算符可以用在 *daemon_lists* 和 *client_lists* 中。 **EXCEPT** 运算符可以嵌套使用。如果控制语言允许使用括号，则 **"a EXCEPT b EXCEPT c"** 可被解析为 **"(a EXCEPT (b EXCEPT c))"**

Shell 命令

如果第一个匹配的访问控制规则包含一个 Shell 命令，则该命令将受制于 **%***letter* 扩展（参阅下一小节）。结果将由一个 **/bin/sh** 子进程执行，并带有连接到 **/dev/null** 的标准的输入、输出和错误。如果不想等待命令完成，可以在命令的结尾指定“&”。

Shell 命令不应依赖于 **inetd** 的 PATH 设置。相反，它们应使用绝对路径名，或者以显式的 **PATH=***whatever statement* 开始。

联机帮助页 *hosts_options*(5) 描述了使用 Shell 命令字段的访问控制语言。

% 扩展

Shell 命令中可以使用以下扩展：

%a(%A) 客户端（服务器）主机地址。

%c 客户端信息： *user@host*, *user@address*, ，一个主机名，或者仅仅是一个地址，这取决于可用信息的多少。

%d 守护程序进程名 (**argv[0]** *value*) 。

%h(%H) 客户端（服务器）主机名或地址（如果主机名不可用）。

%n(%N) 客户端（服务器）主机名（或者 **unknown** 或 **paranoid** ）。

%p 守护程序进程 ID。

%s 服务器信息： *daemon@host*, *daemon@address*, 或者仅仅是一个守护程序名，这取决于有多少可用信息。

%u 客户端用户名（或 **unknown** ）。

%% 单个 % 字符的扩展。

不与任何字母数字 (**A-Za-z0-9**) 或 **!@%-_=+:,./** 字符相匹配的 % 扩展中的字符将被替换为下划线。

服务器端点模式

要按照客户端所连接的网络地址来区分客户端，请使用以下形式的模式：

process_name@*host_pattern*:*client_list*...

当计算机具有不同的 Internet 地址和不同的 Internet 主机名时，可以使用这类模式。服务提供商可以使用此功能来提供 FTP、 GOPHER 或 WWW 归档，这些归档的 Internet 名称甚至可能属于不同的组织。另请参阅 *hosts_options*(5) 中的 **twist** 选项。某些系统可以在一个物理接口上具有多个 Internet 地址。而对于另一些系统，可能需要求助于专用网络地址空间中的 SLIP 或 PPP 伪接口。

host_pattern 遵循与 *client_list* 环境中的主机名和地址相同的语法规则。通常，只有面向连接的服务才会带有服务器端点信息。

客户端用户名查找

如果客户端主机支持 RFC 931 协议或其某个派生协议（TAP、IDENT、RFC 1413），则包装程序可以检索有关连接的所有者的其他信息。客户端用户名信息（如果可用）将与客户端主机名记录在一起，可用于匹配诸如以下模式：

daemon_list:... *user_pattern*@*host_pattern*...

可以在运行时将守护程序包装配置为（在 **/etc/tcpd.conf** 中进行配置）执行由规则驱动的用户名查找（缺省设置）或始终询问客户端主机。对于由规则驱动的用户名查找，根据上述规则，仅当 *daemon_list* 和 *host_pattern* 都匹配时才会进行用户名查找。

用户模式与守护程序进程模式的语法相同，因此可以应用相同的通配符（不支持网络组成员关系）。由于以下限制因素，使用用户名查找时需谨慎考虑：

- 当客户端用户名信息严重不足时，即，当客户端系统已受到安全威胁时，客户端用户名信息是不可信的。一般来讲，只有 **ALL** 和 (**UN**)**KNOWN** 才是有意义的用户名模式。
- 仅对于基于 TCP 的服务，并且仅当客户端主机运行适当的守护程序时，才可能使用用户名查找。在所有其他情况下，结果都将是“未知”。
- 对于非 UNIX 用户来说，用户名查找可能会导致明显的延迟。用户名查找的超时值可以通过 **/etc/tcpd.conf** 来配置。有关详细信息，请参阅 *tcpd.conf*(4) 。

选择性的用户名查找可以缓解上述延迟问题。例如，下面的规则：

daemon_list **: @pcnetgroup ALL@ALL**

将匹配 pc 网络组的成员而不会进行用户名查找，但对于所有其他系统则会执行用户名查找。

检测地址欺骗攻击

许多 TCP/IP 实现的序列号生成器中的缺陷会让入侵者轻而易举地伪装成可信任的主机，并通过，例如远程 Shell 服务，进行入侵。IDENT（RFC931 等）服务可用来检测这类以及其他的主机地址欺骗攻击。

在接受某个客户端请求之前，包装程序可以使用 IDENT 服务来发现该客户端是否真的发送了请求。当客户端主机提供 IDENT 服务时，如果出现负的 IDENT 查找结果（客户端与 **UNKNOWN@***host* 相匹配），则可以明确表

明这是一个主机欺骗攻击。

一个正的 IDENT 查找结果（客户端与 **KNOWN@***host* 相匹配）也不十分可靠。因为入侵者可以同时欺骗客户端连接和 IDENT 查找，尽管这比单纯欺骗客户端连接要难得多。此外，客户端的 IDENT 服务器也有可能撒谎。

注释：IDENT 查找不适用于 UDP 服务。

举例

该语言非常灵活，可以很容易地构造出不同类型的访问控制策略。虽然该语言使用了两个访问控制表，但大多数常用策略可以使用其中的一个表来实现，另一个表只含有很少的内容，甚至完全为空。

阅读下面的示例时，请注意授权表是先于拒绝表扫描的，认识这一点很重要。当找到一个匹配时，搜索即终止；如果未找到任何匹配，则授予访问权限。

示例中使用了主机名和域名。可以对这些示例加以改进，即通过包含地址和（或）网络（或网络掩码）信息来减少临时名称服务器查找失败的影响。

通常关闭的访问控制

在此示例中，缺省情况下访问被拒绝。只有显式授权的主机才被允许进行访问。

该缺省策略（拒绝访问）是通过一个普通的拒绝文件实现的：

/etc/hosts.deny:
  ALL: ALL

这将拒绝向任何主机提供任何服务，除非通过授权文件中的条目授予它们访问权限。

被显式授权的主机列在授权文件中。例如：

/etc/hosts.allow:
  ALL: LOCAL @some_netgroup
  ALL: .foobar.edu EXCEPT terminalserver.foobar.edu

第一条规则允许本地域中的主机（主机名中没有点“.”）以及 *some_netgroup* 网络组中的成员进行访问。第二条规则允许 **foobar.edu** 域中的所有主机（注意 **.foobar.edu** 中的前导点“.”）进行访问，但 *terminalserver.foobar.edu* 除外。

通常开放的访问控制

此时，缺省情况下允许进行访问。只有显式指定主机才是拒绝的服务。

该缺省策略（允许访问）让授权文件变得冗余，因此可以将其忽略。显式非授权的主机列在拒绝文件中。例如：

/etc/hosts.deny:
  ALL: some.host.name, .some.domain

ALL EXCEPT fingerd: other.host.name, .other.domain

第一条规则将拒绝某些主机和域的所有服务。第二条规则仍接受来自其他主机和域的 **finger** 请求。

设置陷阱

下面的示例将接受来自本地域中的所有主机（注意前导点）的 **tftp** 请求。来自任何其他主机的请求将被拒绝；这时系统不会向被拒绝的主机提供它们请求的文件，而是会发送一个 **finger** 探查命令。探查的结果将发送给超级用户。

```
/etc/hosts.allow:
  tftpd: LOCAL, .my.domain

/etc/hosts.deny:
  tftpd: ALL: spawn (/usr/bin/sffinger -l @%h | \
      /usr/bin/mailx -s %d-%h root) &
```

sffinger 命令是 **tcp** 包装程序附带的。它可以限制来自远程 **finger** 服务器发送的数据可能造成的破坏。它提供了比标准 **finger** 命令更好的保护。

%h （客户端主机）和 **%d** （服务名）序列的扩展在前面的“Shell 命令”一节中有介绍。

警告：请不要在 **finger** 守护程序上设置陷阱，除非已经准备好了应对无限的 **finger** 循环。

在网络防火墙系统上，服务陷阱特别有用。典型的网络防火墙仅向外部提供一组有限的服务。所有其他服务都可以像上述 tftp 示例那样设置陷阱。这样便可以获得一个非常优秀的早期警告系统。

诊断信息

问题是通过 **syslogd** 和 **syslog** 守护程序报告的，分为 **info**、 **notice**、**warning** 和 **err** 四个级别。在以下情况下系统会发出错误报告：

- 在主机访问控制规则中发现语法错误，
- 访问控制规则的长度超出了内部缓冲区的容量，
- 访问控制规则未由一个换行符终止，
- *%letter* 扩展的结果会溢出内部缓冲区，
- 不应失败的系统调用失败。

警告

如果某个名称服务器查找超时，则访问控制软件将无法使用该主机名，即使该主机已注册。

域名服务器查找不区分大小写。NIS（以前的 YP）网络组查找区分大小写。

作者

Wietse Venema (wietse@wzv.win.tue.nl)
Department of Mathematics and Computing Science
Eindhoven University of Technology
Den Dolech 2, P.O. Box 513,
5600 MB Eindhoven, The Netherlands

文件

/etc/hosts.allow 被授予访问权限的（守护程序，客户端）对。

/etc/hosts.deny 被拒绝访问的（守护程序，客户端）对。

另请参阅

tcpd(1M) TCP/IP 守护程序包装程序。

tcpdchk(1) 和 *tcpdmatch*(1) 测试程序。

tryfrom(1) 和 *sffinger*(1) TCP 包装实用程序。

名称

hosts_options - 主机访问控制语言扩展

说明

本联机帮助页对 *hosts_access(5)*中所述的可选扩展语言进行了描述。

可扩展语言使用以下格式：

daemon_list **:** *client_list* **:** *option* **:** *option ...*

hosts_access(5) 中对前两个字段进行了描述。简而言之， *daemon_list* 是一个或多个守护程序进程名或通配符的列表。 *client_list* 则是一个或多个将与客户端主机名或地址相匹配的主机名、主机地址、模式或通配符的列表。

规则其余部分是零个或多个选项列表。任何“:”选项内的字符必须以反斜杠“\”加以保护。

选项是“关键字”或“关键字值”形式。选项按指定顺序处理。某些选项可以被 *%letter* 替换。为了与早期版本向后兼容，允许在关键字和值之间使用等号“=”。

日志记录选项

severity mail.info
severity notice

更改将要被记录的事件严重性等级。设备名称（如邮件）是可选的，且不受使用较早 **syslog** 实现的系统所支持。请参阅与设备相关的 *syslog*(3C) 。严重性选项可用于强调或忽略特定的事件。

访问控制选项

allow
deny

分别以 **allow** 和 **deny** 选项允许或拒绝服务。这些选项必须出现在规则结尾处。

allow 和 **deny** 关键字使得在单个文件内遵守所有访问控制规则成为可能，例如在 **hosts.allow** 文件内。举例如下：

仅允许从特定的主机访问：

ALL: .friendly.domain: ALLOW
ALL: ALL: DENY

除了少数麻烦制造者外，允许从所有主机访问：

ALL: .bad.domain: DENY
ALL: ALL: ALLOW

请注意域名模式上的前导点 (.)。

运行其他命令

spawn *shell_command*

执行 *hosts_access(5)* 中所述的 *%letter* 扩展后，在子进程中执行特定的 Shell 命令。此命令通过 **stdin** 、**stdout** 和 **stderr** 连接到空设备执行，因此不会扰乱与客户端主机的通信。例如：

**spawn (/usr/bin/sffinger -l @%h | **
/usr/bin/mailx -s "alert" root) &

在后台子进程中执行 Shell 命令

sffinger -l @%h | mail root

以远程主机名称或地址替换 **%h** 后。

示例用 **sffinger** 命令代替常规 **finger** 命令来限制发送自 finger 服务器的数据可能导致的破坏。 **sffinger** 命令是守护程序包的包装的一部分。它是在常规 **finger** 命令周围的包装，过滤远程主机发送的数据。

twist *shell_command*

执行 *hosts_access(5)* 中所述的 *%letter* 扩展后，以特定的 Shell 命令实例替换当前进程。 **stdin** 、 **stdout** 和 **stderr** 与客户端进程相连接。这种选项必须出现在规则结尾处。

给客户端发送定制弹转消息而不是运行实际 FTP 守护程序：

ftpd : ... : twist /bin/echo 421 Some bounce message

有关另一种与客户端进程进行通信的方式请参阅下面的 **banners** 选项。

在不扰乱命令行数组或其进程环境的情况下运行 /some/other/teneltd：

telnetd : ... : twist PATH=/some/other; exec telnetd

警告：如果使用 UDP 服务，请不要转换到使用标准 I/O 或 **read()/write()** 例行程序与客户端进程通信的命令。UDP 需要其他 I/O 原语。

网络选项

keepalive

使服务器定期给客户端发送消息。当客户端不响应时，则认为连接中断。可以用 **keepalive** 选项使用户虽关闭机器但仍保持与服务器相连接。 **keepalive** 选项不能用于数据报 (UDP) 服务。

linger *number_of_seconds*

指定内核在服务器进程关闭连接后传递尚未被传递的数据所需要的时间。

用户名查找选项

rfc931 [*timeout_in_seconds*]

用 RFC 931（TAP、IDENT、RFC 1413）协议查找客户端用户名。基于传输服务而不是 TCP 服务的情况下，这种选项将以无提示方式忽略。它需要客户端系统运行一个遵循 RFC 931 规范的守护程序（IDENT 等等），可能导致与非 UNIX 客户端的连接出现显著延迟。超时时间可通过配置文件 **/etc/tcpd.conf** 进行调整。如果没有指定超时时间或指定超时时间无效，用户名查找将无法进行。

其他选项

banners */some/directory*

在 */some/directory* 中查找与守护程序进程同名的文件（例如，输入 **telnetd** 查找 Telnet 服务），并将其内容复制到客户端。以回车键取代换行符，同时对 *%letter* 序列进行扩展（请参阅 *hosts_access(5)*）。

将文本发送至服务协议中指定的客户端时，标题选项不会添加任何服务特定的字符。要成功使用此选项，除实际文本之外，文件还必须包含必需的协议参数。

例如，在 **ftpd** 服务中，标题文件中的各行不会自动将 FTP RFC 959 中定义的状态代码 (220-) 添加为前缀。因此，如果要将下列文本发送至 FTP 客户端：

```
This is a sample Welcome text to demonstrate the banners
option in tcpd.
```

建议您添加协议特定的如下响应代码：

```
220-This is a sample Welcome text to demonstrate the banners
220-option in tcpd.
```

对于 **rlogind** 服务，必须在 **rlogind** 标题文件的开端放置一个空字符 (**\0**)，如下所示：

```
# echo "\0This is a sample Welcome text to demonstrate \
  the banners" > rlogind
# echo "option in tcpd." >> rlogind
```

可以使用 **/usr/examples/tcpd/Banners.Makefile** 文件生成多个服务的标题。有关详细信息，请参考 **/usr/examples/tcpd/Banners.Makefile**。

警告：仅为面向连接的 (TCP) 网络服务提供横幅。

nice [*number*]

更改进程 nice 值（缺省值为 10）。在其他进程中指定一个正值占用更多 CPU 资源。

setenv *name value*

在进程环境中指定一对（名称、值）。值被 *%letter* 扩展且可能包含空格（但去除前导和结尾空格）。

警告：许多网络守护程序在启动登录或 Shell 进程前会对其环境进行重置。

umask 022

比如在 Shell 中创建 **umask** 命令。022 **umask** 命令禁止创建组及全局写入权限文件。 **umask** 参数必须是八进制数。

user *someuser* 或 **user** *someuser.somegroup*

假定“someuser”用户（或用户“someuser”，组“somegroup”）的优先权。第一种格式可用于 **inetd** 实现，从而在超级用户权限下运行所有服务。第二种格式仅可用于需要专门组权限的服务。

诊断信息

通过 **syslogd**，即 **syslog** 守护程序报告问题，且问题分为 **info**、**notice**、**warning** 和 **err** 级别。当在访问控制规则中查找到语法错误时，错误将报告至 **syslog** 守护程序；这时进程会忽略其他选项并拒绝服务。

作者

Wietse Venema (wietse@wzv.win.tue.nl)
Department of Mathematics and Computing Science
Eindhoven University of Technology
Den Dolech 2, P.O. Box 513,
5600 MB Eindhoven, The Netherlands

另请参阅

hosts_access(5)、缺省的访问控制语言。

名称

intr_strobe_ics_pct - 限制处理器可用于中断环境的时间百分比

值

保证安全

100

缺省值

80

允许值

80 - 100

将 **intr_strobe_ics_pct** 设置为 **100** ，可关闭其功能。

建议值

100

说明

intr_strobe_ics_pct 指定了系统对处理器可用于中断环境的时间百分比的限制。如果此可调参数设置的值小于 **100**，则内核不允许 I/O 中断占用处理器的时间超出所设定的限制值，以便让线程与其他低优先级中断可获得执行时间。当系统处于繁重的 I/O 中断负荷情况时，这是十分重要的，某些线程，例如服务保护环境中的心跳计时器，需要在有限的延迟时间里运行。

在正常工作的系统中，可调参数 **intr_strobe_ics_pct** 的值应为 **80** 。

如果将 **intr_strobe_ics_pct** 设置为小于 **100** ，则如下的类似消息可能会出现在消息缓冲区。

An interrupt context threshold of #% was exceeded on processor # within the last # minutes. The current value is #.

消息缓冲区可以通过 **dmesg** 或 **syslog** 读取。此消息表明处理器上所占用的时间百分比超过了指定限制，同时已采取了更正操作。

更改此可调参数的人员

仅有 HP 现场工程师能更改此可调参数值。

更改限制

对此可调参数的更改立即生效。

应该何时降低此可调参数的值

仅当繁重的 I/O 中断负荷引起时间关键进程超时，才应降低此可调参数的值。

降低此值的负面影响

降低此可调参数的值能为线程环境提供更多的执行时间。但是，整个系统的处理能力可能会降低。

应该同时更改的其他可调参数值

此可调参数与其他可调参数无关。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

intr_strobe_ics_pct 由 HP 开发。

另请参阅

kctune(1M)、 kcweb(1M)。

名称

inttypes - 固定大小的整型数据类型

概要

#include <inttypes.h>

说明

此头文件定义了各种大小的整型数据类型。使用在此头文件中定义的数据类型，开发者可以确保数据类型在不同的系统中具有相同的属性及行为。

由于本手册页中定义的所有整数大小不需要所有的实现支持，因此了解当前实现是否支持一个整数的特定大小的正确方式是，测试定义其最大值的符号。例如，如果 **#ifdef UINT64_MAX** 测试失败，那么此实现就不支持 64 位无符号整数。

<inttypes.h> 头文件中包含 **<stdint.h>** 头文件。

此头文件定义了下列文件类型。

imaxdiv_t 结构体类型，是 **imaxdiv()** 函数返回的值的类型

下列宏可用作具有 **printf()** 系列功能的格式选项（请参阅 *printf*(3S)）。这些宏用于为本手册页中前面定义的整型数据类型选择正确的格式选项。

PRId8 **d** 输出 int8_t 的格式设置选项
PRId16 **d** 输出 int16_t 的格式设置选项
PRId32 **d** 输出 int32_t 的格式设置选项
PRId64 **d** 输出 int64_t 的格式设置选项
PRIdMAX **d** 输出 intmax_t 的格式设置选项

PRIdFAST8 **d** 输出 int_fast8_t 的格式设置选项
PRIdFAST16 **d** 输出 int_fast16_t 的格式设置选项
PRIdFAST32 **d** 输出 int_fast32_t 的格式设置选项
PRIdFAST64 **d** 输出 int_fast64_t 的格式设置选项
PRIdFAST **d** 输出 intfast_t 的格式设置选项

PRIdLEAST8 **d** 输出 int_least8_t 的格式设置选项
PRIdLEAST16 **d** 输出 int_least16_t 的格式设置选项
PRIdLEAST32 **d** 输出 int_least32_t 的格式设置选项
PRIdLEAST64 **d** 输出 int_least64_t 的格式设置选项

PRIi8 **i** 输出 int8_t 的格式设置选项
PRIi16 **i** 输出 int16_t 的格式设置选项
PRIi32 **i** 输出 int32_t 的格式设置选项

PRIi64 **i** 输出 int64_t 的格式设置选项

PRIiFAST8 **i** 输出 int_fast8_t 的格式设置选项
PRIiFAST16 **i** 输出 int_fast16_t 的格式设置选项
PRIiFAST32 **i** 输出 int_fast32_t 的格式设置选项
PRIiFAST64 **i** 输出 int_fast64_t 的格式设置选项

PRIiLEAST8 **i** 输出 int_least8_t 的格式设置选项
PRIiLEAST16 **i** 输出 int_least16_t 的格式设置选项
PRIiLEAST32 **i** 输出 int_least32_t 的格式设置选项
PRIiLEAST64 **i** 输出 int_least64_t 的格式设置选项

PRIu8 **u** 输出 uint8_t 的格式设置选项
PRIu16 **u** 输出 uint16_t 的格式设置选项
PRIu32 **u** 输出 uint32_t 的格式设置选项
PRIu64 **u** 输出 uint64_t 的格式设置选项
PRIuMAX **u** 输出 uintmax_t 的格式设置选项

PRIuFAST8 **u** 输出 uint_fast8_t 的格式设置选项
PRIuFAST16 **u** 输出 uint_fast16_t 的格式设置选项
PRIuFAST32 **u** 输出 uint_fast32_t 的格式设置选项
PRIuFAST64 **u** 输出 uint_fast64_t 的格式设置选项
PRIuFAST **u** 输出 uintfast_t 的格式设置选项

PRIuLEAST8 **u** 输出 uint_least8_t 的格式设置选项
PRIuLEAST16 **u** 输出 uint_least16_t 的格式设置选项
PRIuLEAST32 **u** 输出 uint_least32_t 的格式设置选项
PRIuLEAST64 **u** 输出 uint_least64_t 的格式设置选项

PRIo8 **o** 输出 int8_t 的格式设置选项
PRIo16 **o** 输出 int16_t 的格式设置选项
PRIo32 **o** 输出 int32_t 的格式设置选项
PRIo64 **o** 输出 int64_t 的格式设置选项
PRIoMAX **o** 输出 intmax_t 的格式设置选项

PRIoFAST8 **o** 输出 int_fast8_t 的格式设置选项
PRIoFAST16 **o** 输出 int_fast16_t 的格式设置选项
PRIoFAST32 **o** 输出 int_fast32_t 的格式设置选项
PRIoFAST64 **o** 输出 int_fast64_t 的格式设置选项

PRIoFAST **o** 输出 intfast_t 的格式设置选项

PRIoLEAST8 **o** 输出 int_least8_t 的格式设置选项
PRIoLEAST16 **o** 输出 int_least16_t 的格式设置选项
PRIoLEAST32 **o** 输出 int_least32_t 的格式设置选项
PRIoLEAST64 **o** 输出 int_least64_t 的格式设置选项

PRIx8 **x** 输出 int8_t 的格式设置选项
PRIx16 **x** 输出 int16_t 的格式设置选项
PRIx32 **x** 输出 int32_t 的格式设置选项
PRIx64 **x** 输出 int64_t 的格式设置选项
PRIxMAX **x** 输出 intmax_t 的格式设置选项

PRIxFAST8 **x** 输出int_fast8_t 的格式设置选项
PRIxFAST16 **x** 输出int_fast16_t 的格式设置选项
PRIxFAST32 **x** 输出int_fast32_t 的格式设置选项
PRIxFAST64 **x** 输出int_fast64_t 的格式设置选项
PRIoFAST **x** 输出 intfast_t 的格式设置选项

PRIxLEAST8 **x** 输出 int_least8_t 的格式设置选项
PRIxLEAST16 **x** 输出 int_least16_t 的格式设置选项
PRIxLEAST32 **x** 输出 int_least32_t 的格式设置选项
PRIxLEAST64 **x** 输出 int_least64_t 的格式设置选项

PRIX8 **X** 输出 int8_t 的格式设置选项
PRIX16 **X** 输出 int16_t 的格式设置选项
PRIX32 **X** 输出 int32_t 的格式设置选项
PRIX64 **X** 输出 int64_t 的格式设置选项

PRIXFAST8 **X** 输出 int_fast8_t 的格式设置选项
PRIXFAST16 **X** 输出 int_fast16_t 的格式设置选项
PRIXFAST32 **X** 输出 int_fast32_t 的格式设置选项
PRIXFAST64 **X** 输出 int_fast64_t 的格式设置选项

PRIXLEAST8 **X** 输出 int_least8_t 的格式设置选项
PRIXLEAST16 **X** 输出 int_least16_t 的格式设置选项
PRIXLEAST32 **X** 输出 int_least32_t 的格式设置选项
PRIXLEAST64 **X** 输出 int_least64_t 的格式设置选项

下列宏可用作具有 **scanf()** 系列功能的格式选项（请参阅 *scanf*(3S)）。这些宏用于为本手册页中前面定义的整型

数据类型选择正确的格式选项。

SCNd16 **d** 扫描 int16_t 的格式设置选项
SCNd32 **d** 扫描 int32_t 的格式设置选项
SCNd64 **d** 扫描 int64_t 的格式设置选项
SCNdMAX **d** 扫描 intmax_t 的格式设置选项

SCNi16 **i** 扫描 int16_t 的格式设置选项
SCNi32 **i** 扫描 int32_t 的格式设置选项
SCNi64 **i** 扫描 int64_t 的格式设置选项
SCNiMAX **i** 扫描 intmax_t 的格式设置选项

SCNu16 **u** 扫描 uint16_t 的格式设置选项
SCNu32 **u** 扫描 uint32_t 的格式设置选项
SCNu64 **u** 扫描 uint64_t 的格式设置选项

SCNo16 **o** 扫描 int16_t 的格式设置选项
SCNo32 **o** 扫描 int32_t 的格式设置选项
SCNo64 **o** 扫描 int64_t 的格式设置选项
SCNoMAX **o** 扫描 intmax_t 的格式设置选项

SCNx16 **x** 扫描 int16_t 的格式设置选项
SCNx32 **x** 扫描 int32_t 的格式设置选项
SCNx64 **x** 扫描 int64_t 的格式设置选项
SCNxMAX **x** 扫描 intmax_t 的格式设置选项

SCNdFAST **d** 扫描 intfast_t 的格式设置选项
SCNiFAST **i** 扫描 intfast_t 的格式设置选项
SCNoFAST **o** 扫描 intfast_t 的格式设置选项
SCNxFAST **x** 扫描 intfast_t 的格式设置选项

注释

printf() 系列功能的格式选项均以 **PRI** 开头，而 **scanf()** 系列功能的格式选项则均以 **SCN** 开头。这些格式字符串不可互换。

举例

以下示例显示如何使用其中一种具有 **printf()** 功能的打印格式选项。

uint64_t u;

...

```
printf("u = %016" PRIx64 "\n", u);
```

作者

inttypes.h 由 HP 开发。

文件

/usr/include/inttypes.h

另请参阅

imaxdiv(3)、 printf(3S)、 scanf(3S)、 standards(5)、 stdint(5)。

名称

ioctl - 常规设备控制命令

概要

#include <sys/ioctl.h>

ioctl(fildes, request, arg)

int fildes, request;

说明

ioctl(2) 系统调用为开放设备提供控制。这个包含文件描述了用于 *ioctl*(2) 、具有常规特点的 *requests* 和 *arguments*。有关单个请求如何影响任何特定设备的详情，请参阅《HP-UX 参考手册》第 7 节中相应的设备手册条目。如果设备不支持读写控制请求，则返回 [EINVAL] 。

FIONREAD 以其地址为 *arg* 的长整型数，返回从设备文件立即可读的字符数目。

FIOSSAIOSTAT 对于那些支持此命令的字符设备文件来说，如果地址为 *arg* 的整数非零，则启用系统异步 I/O；也就是说，只要设备相关文件事件发生，就可以将 **SIGIO** 发送到当前 **FIOSSAIOOWN** 指定的进程（请参阅下文）。如果 **FIOSSAIOOWN** 没有指定进程，那么可以将 **SIGIO** 发送到第一个进程以打开设备文件。

如果指定的进程已经退出，那么 **SIGIO** 信号不会被发送到任何进程。

如果地址为 *arg* 的整数是 0，则禁用系统异步 I/O。

FIOGSAIOSTAT 如果启用系统异步 I/O，对那些支持此命令的字符设备文件，地址为 *arg* 的整数，则设置为 1。否则，地址为 *arg* 的整数将设置为 0。

FIOSSAIOOWN 对那些支持此命令的字符设备文件，设置进程 ID 来接收 **SIGIO** 信号，该信号带有系统异步 I/O，面向地址为 *arg* 的整数值。拥有相应特权的用户可以指定任意进程接收 **SIGIO** 信号。如果超级用户没有作出请求，则只有通过调用进程来指定该调用进程本身或另一进度（此进度的真实或已保存的有效用户 ID 与此进度的真实的或有效的用户 ID 相匹配），或某个调用进程的子进程来接收 **SIGIO** 信号。如果找不到与地址为 arg 的整数指定相对应的进程，则 **errno** 设置为 [ESRCH] 显示调用失败。如果超级用户没有作出请求，且调用进程尝试指定一个进程而不是其自身或 (1) 其他实际、或已保存的有效用户 ID 与其自身实际、或已保存的有效用户 ID 相匹配的进程、或 (2) 非调用进程的子进程，则调用失败， **errno** 设置为 [EPERM] 。

如果指定的进程随后退出， **SIGIO** 信号不会被发送至任何进程。

当打开设备文件时，缺省模式为将执行打开的进程设置为接收 **SIGIO** 信号。

FIOGSAIOOWN 对那些支持此命令的字符设备文件，将地址为 *arg* 的整数设置到指定接收 **SIGIO** 信号的进程 ID。

FIOSNBIO 对那些支持此命令的字符设备文件，如果地址为 *arg* 的整数非零，则启用非阻塞 I/O；也就是说，后续的读取和写入设备文件被处理为非阻塞方式（请参阅下文）。如果地址为 *arg* 的整数是 0，则禁用非阻塞 I/O。

对于读取，非阻塞 I/O 禁止阻塞所有对设备的读取请求，而不管请求是成功还是失败。这样的读取请求通过下列三种方式之一完成：

- 如果有足够的数据满足整个请求，则在读取了所有数据后，读取成功完成并返回其读取的字节数；
- 如果没有足够的数据满足整个请求，则在读取了尽可能多的数据后，读取成功完成并返回其读取的字节数；
- 如果没有足够的数据，则读取失败且 **errno** 设置为 [EWOULDBLOCK] 。

对于写入，非阻塞 I/O 禁止阻塞所有对设备文件的写入请求， 而不管该请求是成功还是失败。这样的写入请求通过下列三种方式之一完成：

- 如果系统有足够的空间缓冲所有数据，则在写入所有数据后，写入成功完成并返回其写入的字节数；
- 如果缓冲区没有足够的空间写出整个请求，则在写入尽可能多的数据后，写入成功完成，并返回其可写入的字节数；
- 如果缓冲区没有空间，则写入失败且 **errno** 设置为 [EWOULDBLOCK] 。

为了禁止非阻塞 I/O 影响 **O_NDELAY** 标记（请参阅 *open*(2) 和 *fcntl*(2) ）， **O_NDELAY** 的功能始终取代非阻塞 I/O 的功能。这意味着，如果设置了 **O_NDELAY** ，驱动程序根据 **O_NDELAY** 定义执行读取请求。当没有设置 **O_NDELAY** 时，应用非阻塞 I/O 定义。

打开设备文件时，缺省为禁用非阻塞 I/O。

FIOGNBIO 对那些支持此命令的字符设备文件，如果启用非阻塞 I/O，则地址为 *arg* 的整数设置为 1。否则，地址为 *arg* 的整数设置为 0。

警告

FIOSSAIOSTAT 类似于 4.2 BSD **FIOASYNC** ，前者新增了安全性规定。

FIOGSAIOSTAT 源自 HP，它补充了 **FIOSSAIOSTAT** ，并允许保存和恢复系统异步 I/O TTY 状态，以便用于 BSD 样式的作业控制。

FIOSSAIOOWN 类似于 4.2 BSD **FIOSETOWN** ，前者新增了安全性规定。

FIOGSAIOOWN 类似于 4.2 BSD **FIOGETOWN** 。还应注意该功能的 4.2 BSD 版本与 HP-UX 版本的区别，前者使用进程组，而后者仅使用进程。

FIOSNBIO 类似于 4.2 BSD **FIONBIO** ，区别在于前者不会干预 AT&T **O_NDELAY** *open* 和 *fcntl* 标记。

FIOGNBIO 源自 HP，它补充了 **FIOSNBIO** ，并允许保存和恢复非阻塞的 I/O TTY 状态，以便用于 BSD 样式的作业控制。

另请参阅

ioctl(2)、 arp(7)、 socket(7)。

《HP-UX 参考手册》第 7 节

名称

ipmi_watchdog_action - 设置在 IPMI 监视程序计时器过期时执行的操作

值

保证安全

0 （无操作）

缺省值

0 （无操作）

允许值

0 不执行任何操作

1 硬重置系统（或非单元式系统上的分区）

2 关闭电源（仅对非单元式系统有效）

3 关闭系统电源后重新打开（仅对非单元式系统有效）

建议值

使用的值取决于系统无法向前运行时，系统管理员希望发生什么情况。

说明

ipmi_watchdog_action 可调参数可以在带有 IPMI（智能平台管理接口）监视程序计时器的系统上设置监视程序计时器过期操作。

要判断系统是否具有此功能，请使用 **ioscan -fnkC ipmi** 。如果列出了 **ipmi** ，则表示支持 IPMI 监视程序计时器。

HP 的 Itanium 产品系列系统和最新的 PA RISC 系统将 IPMI 用作系统管理基础结构的一部分。操作系统挂起或不响应时，IPMI 将使用“监视程序计时器”来检测条件。发生此种情况时，将自动执行使用 **ipmi_watchdog_action** 可调参数配置的操作。

例如，如果该参数设置为 **1** ，那么在操作系统挂起或不响应时，系统（或单元式系统上的分区）将会硬重置。

更改此可调参数的人员

在系统停止向前运行的情况下，希望系统执行某项操作的系统管理员。

更改限制

无。此可调参数为动态的。

应该同时更改的其他可调参数值

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在

用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

ipmi_watchdog_action 由 HP 开发。

另请参阅

kctune(1M)、 sam(1M)、 gettune(2)、 settune(2)。

名称

kconfig - 内核配置命令简介

说明

HP-UX 包含一组用来查看和修改 HP-UX 内核配置的命令。这些命令包括:

kconfig 处理整个内核配置

kcmodule 处理内核模块

kctune 处理内核可调参数 (tunable)

kcpath 检索内核文件的路径名

kclog 搜索并显示内核配置日志文件

mk_kernel 由系统文件构建内核配置

内核配置

内核配置是指控制 HP-UX 内核的行为和内容的一组数据。系统管理员可以保存任意多组内核配置，并且可以随时加载其中的任何一组配置。内核配置包含了对所用模块的选择（使用 **kcmodule** ）以及对可调参数值的选择（使用 **kctune** ）。

缺省情况下，这些命令将影响当前运行系统的状态。如果这些命令带有 **-c** *config* 选项，则会影响所保存的名为 *config* 的内核配置。

使用 **kconfig -s** 可以保存当前运行的内核配置。使用 **kconfig -l** 可以载入某个保存的配置。这会导致当前运行系统的状态根据该保存的配置发生相应变化。使用 **kconfig -n** 可以标记出系统下一次引导时所要使用的某个保存的配置。这不会改变当前运行系统的状态，但会在系统重新引导时载入指定的已保存配置（请参阅下文的“引导行为”一节）。

所保存的内核配置的名称必须以字母开头，并且只能包含字母、数字和下划线 (_)；长度不能超过 32 个字符。名称区分大小写。

备份配置

系统维护着一个称为 **backup** 的已保存配置，可用来在发生配置错误时进行恢复。根据选定的备份行为，系统可以在根据请求更改配置之前，立即自动地将当前允许的配置保存到 **backup** 。备份行为可使用 **kconfig** 、 **kcmodule** 或 **kctune** 命令的 **-b** *behavior* 选项进行设置。可识别的备份 行为包括:

yes 始终在变更之前更新 **backup** 配置。

once 在进行当前的变更之前，更新 **backup** 配置。对于此后的变更，将询问是否更新。

no 在进行当前的变更之前，不更新 **backup** 配置。对于此后的变更，将询问是否更新。

disable 永远不在变更之前更新 **backup** 配置。

这些行为可以缩写为一个字符。为了兼容以前的版本， **-B** 可作为 **-b yes** 的别名， **-K** 可作为 **-b no** 的别名。在将来的版本中将删除这些别名。

在每次引导后，缺省的备份行为将询问是否在每次变更之前更新 **backup** 配置。非交互方式进行的变更将假定响应为“no”。

动态变更和静态变更

缺省情况下，各内核配置工具会将配置变更应用到当前运行系统中，从而立即更改当前运行系统的行为。通过向使用 **kconfig** 、 **kcmodule** 或 **kctune** 命令进行的变更指定 **-h** 选项，系统管理员可以覆盖此缺省值。该选项可让所做的变更在系统下一次引导时应用。 HP 建议仅当准备不久后即重新引导系统时才使用此选项。如果在变更后数月内都未进行重新引导，并且管理员又忘记了这一变更请求，那么他（或她）可能会对这些变更感到很困惑。

某些配置变更必须在系统重新引导后才能应用。在系统重新引导之前，即使未指定 **-h** 选项，也不会应用这些变更。在这种情况下，系统会输出一条警告消息。

如果在某个内核配置命令的单个调用中多次请求执行配置变更，且其中任一变更要求重新引导系统，那么在重新引导之前，系统不会应用任何请求的变更。尤其是，如果使用 **kconfig -l** 载入某个保存的内核配置，并且该配置在重新引导之前无法使用，则当前运行系统的状态不会发生改变，而指定的内核配置被标记为在下次引导时使用。

如果某项配置变更要在下次引导后才能使用，而针对同一配置设置的后续变更可以立即生效，则可以立即生效的变更将被优先应用。第一项变更将在下次引导时生效。在这种情况下，系统会输出警告。

对于将替换当前运行的整个配置（例如， **kconfig -i** （导入）、 **kconfig -l** （加载）或 **kconfig -n** （下次引导））的变更，将导致任何在下次引导时才会生效的变更被忽略。

重新引导时，系统将保留对当前运行系统所做的变更。在进行变更或载入某个保存的内核配置之前，它们将一直有效。

引导行为

引导系统时，管理员可以在引导命令行上指定某个已保存的内核配置的名称（请参阅 *hpux*(1M) 和 *hpux.efi*(1M)）。指定后，系统会在引导过程中载入该内核配置。

如果未在引导命令行上指定内核配置，系统将搜寻任何标记为下次引导时使用的内核配置（通过 **kconfig -n** 、 **kconfig -l** 或 **kconfig -i** 命令标记）。如果找到任何这类配置，系统将在引导过程中载入该配置。

如果未在引导命令行上指定内核配置，也未找到任何标记为下次引导时使用的配置，系统将使用在重新引导之前所使用的相同的配置进行引导。如果该配置具有任何要在重新引导时应用的变更（由于在重新引导之前无法应用，或者是使用了 **-h** 选项），则系统将在重新引导过程中应用这些变更。

如果内核配置无法正确引导，则可以通过引导 **backup** 配置和（或）使用“保证安全引导”标记（在基于 Itanium® 的系统上为 **-tm** ，在 PA-RISC 系统上为 **-f0x40000** ）尝试恢复。有关详细信息，请参阅 *hpux*(1M) 和 *hpux.efi*(1M) 。

系统文件

HP-UX 以前版本的用户可能习惯于将内核配置选择保存在一个名为 **/stand/system** 的文本文件中。这类文件称为“系统文件”。系统会自动维护一个针对当前运行的内核配置的系统文件。该文件位于 **/stand/system** 下。此外，系统还会针对每个保存的内核配置自动维护一个系统文件。这些文件位于 **/stand/***config***/system** 下，其中 *config* 是所保存配置的名称。一旦使用某个内核配置命令对内核配置（保存的配置或当前配置）进行了更改，所对应的系

统文件即被自动重写以反映所作的变更。也可以按需使用 **kconfig -e** 为任何配置生成系统文件。系统文件的格式如 *system*(4) 中所述。

可以在文本编辑器中修改某个系统文件，然后运行 **kconfig -i** 来更改配置。该命令将读取系统文件并根据系统文件的内容对内核配置进行相应修改。 **mk_kernel** 也可以读取系统文件并修改内核配置。保留该命令主要是为了保持与之前发行的 HP-UX 的兼容。

注释： 某些配置变更无需使用内核配置命令（例如，可以通过直接调用 **settune()** 或 **modload()** 系统调用进行更改）。在这种情况下，系统文件不会自动更新。请确保在使用它们之前手动进行更新，或者使用 **kconfig -e** 重新创建。

注释： 请避免在系统文件中加入注释。因为每次进行内核配置变更时都要重新创建系统文件，该过程不会保留任何注释。

系统文件可用来将内核配置传送给其他系统。为此，可以在源计算机上使用 **kconfig -e** 将配置导出到一个系统文件中。然后将此文件移至一台或多台目标计算机上，并使用 **kconfig -i** 将该系统文件导入到目标计算机上的配置中。目标计算机必须安装有相同的内核文件集，否则导入操作可能会失败。可以使用 **-V** 标记来确保目标计算机安装有完全相同版本的内核文件集。

如果对当前运行的内核配置进行了要在重新引导后才能生效的变更，则这些变更会反映在系统文件 **/stand/system** 中。

日志文件

内核配置命令维护着一个日志文件，其中描述了所做的所有内核配置变更。该日志文件位于 **/var/adm/kc.log** 中。可以使用 **kclog** 命令来搜索和查看日志文件，或者输入并非对应于配置变更的条目。

使用任何命令更改配置时，都可以指定一个 **-C** *comment* 选项。此时，命令会在描述所做更改的日志文件条目中包含指定的 *comment* 。请注意，通常必须用引号将 *comment* 括起来，以避免 Shell 对它进行解释。

某些配置变更无需使用内核配置命令。对于这类变更，系统不会生成日志文件条目。

日志文件的格式可能会发生变化但并无相应通知。程序必须使用 **kclog** 命令从日志文件中检索条目，而不能尝试对文件格式进行分析。

诊断信息

内核配置命令输出的所有错误、警告和注释消息都有编号。但为了美观，通常并不会显示消息号。要显示消息号，可将环境变量 **KC_SHOW_MSGIDS** 设置为 1。

分析输出

大多数内核配置命令都会生成表格式的输出结果，用来描述配置的详细情况。这种输出便于人阅读，但却不利于脚本和应用程序对其进行分析。此外，表格输出格式可能随时发生变化：例如，在不同类型的系统之间或不同版本的 HP-UX 之间的变化。

基于这些原因，每个生成这类输出的内核配置命令都带有一个 **-P** 选项，它可以改变输出格式。 **-P** 格式旨在让分析过程更容易进行，并且保证格式不会发生更改。 HP 不支持分析内核配置命令的输出的应用程序和脚本，除非它们使用 **-P** 选项。

-P 选项必须后跟一列由逗号分隔的字段名。每个内核配置命令都支持一组不同的字段名；有关名称列表，请参考相应命令的联机帮助页。各字段名必须出现在一个参数中，因此字段名列表中不能包含任何空格。例如，

kcmodule -P name,state,desc

内核配置命令输出的组成如下：首先是描述一个对象的一系列行，然后是一个空行，然后是描述另一个对象的一系列行，然后又是一个空行；如此下去，直至所有的对象都描述完毕。系列行中的每一行都包括一个所描述对象的字段名、一个制表符 (ASCII 9) 以及该字段的值。行的显示顺序与请求的顺序相同。因此，上述命令可能生成如下输出：

```
name    module1
state   loaded
desc    This is the first sample module.

name    module2
state   unused
desc    This is a different sample module.
```

某些字段可能在一个对象中多次出现，也可能根本就不出现。这种现象会出现在字段的说明中。例如，命令：

kcmodule -P name,state,depend

可能生成如下输出：

```
name    module1
state   loaded

name    module2
state   unused
depend  module1
depend  module4
```

这表明 **module1** 不具有依存关系，而 **module2** 依存于其他两个模块。

可以随时添加新字段，但除非在 **-P** 选项中指定，否则输出中不会包含这些字段。字段不能删除。所以将来的开发工作中可能会给出一个没有意义的字段，但这种情况非常少见。在这种情况下，仍接受该字段名，但输出中将省略对应的行。

另请参阅

hpux(1M)、 hpux.efi(1M)、 kclog(1M)、 kcmodule(1M)、 kconfig(1M)、 kcpath(1M)、 kctune(1M)、 mk_kernel(1M)、 modload(2)、 settune(2)、 system(4)。

《HP-UX System Administrator’s Guide: Configuration Management》，可从 **http://docs.hp.com** 上获得。

名称

Kerberos - Kerberos 系统简介

说明

Kerberos 系统可对网络环境中的单个用户进行验证。用户通过 Kerberos 验证后，可以使用网络实用程序，例如：**rlogin** 、 **rcp** 和 **remsh** ，而且不必向远程主机提交口令，也不必编辑和使用 **.rhosts** 文件。注意，只有当用户使用的远程计算机支持 Kerberos 系统时，这些实用程序才可以在没有口令的情况下进行工作。

如果用户输入了用户名，但远程计算机不是 Kerberos 系统，则用户将得到以下消息：

> **Principal unknown (Kerberos) you haven't been registered as a Kerberos user**.

当遇到上述消息时，用户将必须访问系统管理员。

Kerberos 名称通常包含三个部分。第一部分为 *primary* ，通常为一个用户或服务的名称。第二部分是 *instance* ，在用户情况下通常为空。某些用户可能具有特权实例，例如“root”或“admin”。在服务情况下，实例是其运行所在的计算机的完全限定名称；也就是说，运行在计算机 ABC 上的 **rlogin** 服务，与运行在计算机 XYZ 上的 **rlogin** 服务不同。Kerberos 名称的第三部分是 *realm* 。领域对应于为主体提供验证的 Kerberos 服务。

当写入一个 Kerberos 名称时，主体名称与 *instance* （如果不为空）由斜线 (/) 分隔，后跟领域（如果不是本地领域），领域前加上 @ 符号。下面是有效的 Kerberos 名称示例：

> **david**
>
> **jennifer/admin**
>
> **joeuser@BLEEP.BOM**
>
> **cbrown/root@FUBAR.ORG**

当用户进行 Kerberos 验证时，可获取一个初始 Kerberos 凭证。Kerberos 凭证是一个提供验证的加密协议消息。Kerberos 将此凭证用于网络实用程序，例如： **rlogin** 和 **rcp** 。凭证事务处理透明化，因此用户不必担心管理问题。

但是，应注意凭证会过期。具有权限的凭证（例如那些具有“root”实例的凭证）会在几分钟后过期，而对于具有更多普通权限的凭证，其有效期可能是几小时或一天，这取决于 Kerberos 服务器配置。如果用户的登录会话超出了生命周期限制，则用户将必须重新进行 Kerberos 验证，以获取新的凭证。用户可使用 **kinit** 命令对自身进行重新验证。

如果用户使用 **kinit** 命令得到了凭证，那么在结束登录会话前，用户应确保使用 **kdestroy** 命令消除用户凭证。应将 **kdestroy** 命令放在 **.logout** 文件中，以便用户注销时自动消除凭证。有关 **kinit** 和 **kdestroy** 命令的详细信息，请参阅 *kinit*(1) 和 *kdestroy*(1) 。

Kerberos 凭证可以转发。要转发凭证，用户在使用 **kinit** 命令时，必须要请求 *forwardable* 凭证。一旦用户拥有可转发的凭证，大多数 Kerberos 程序有一个命令行选项，可以将凭证转发到远程主机。

目前，Kerberos 支持可用于以下网络服务： **rlogin** 、 **remsh** 、 **rcp** 、 **telnet** 、 **ftp** 和 **ssh** 。

作者

Kerberos 由麻省理工学院开发，开发人包括 MIT Project Athena/Digital Equipment Corporation 的 Steve Miller 和 MIT Project Athena 的 Clifford Neuman。

另请参阅

kdestroy(1)、 kinit(1)、 klist(1)、 kpasswd(1)、 libkrb5(3)、 krb5.conf(4)。

名称

krs - 内核注册服务 KRS

说明

KRS 是一种内核专用机制，用于维护结构化数据。内核中的子系统可以使用 KRS 来维护静态或动态数据。此数据可以是易失性的，也可以是通过系统重新引导而持久不变的。

作者

krs 由 HP 开发。

另请参阅

krs_flush(1M)、 krsd(1M)。

名称

ksi_alloc_max - 可被分配的队列信号的系统级限制

值

缺省

NPROC * 8

允许值

32 到 **maxint (0x7fffffff)**

建议

NPROC * 8

说明

ksi_alloc_max 是对可分配和使用的队列信号的数量的系统级限制。 *ksi* 代表“内核信号信息”并使用有关队列信号信息标识条目。每个队列信号一个条目。队列信号由 **sigqueue** 系统调用、计时器到期、POSIX 实时消息队列及异步 I/O 使用。用户生成的信号（通过 **kill** 和 **raise** ）不会排队。

应该由谁来更改此可调参数?

任何人。

对于更改的限制

无。此可调参数为动态的。

何时应增加此可调参数的值?

频繁或大量使用那些使用队列信号的设备时，可能需要增加此值。由于缺省值是基于进程数量而使用时则是基于线程，因此使用队列信号的每个进程的大量线程将要求增加此可调参数的值。当应用程序使用队列信号返回 [EAGAIN] 时，应该增加此可调参数。

增加此值的副作用是什么?

内存使用增加，但仅在使用时增加。每个已分配的条目都是 96 字节。

何时应降低此可调参数的值?

仅要控制应用程序使用队列信号时。

降低此值的副作用是什么?

如果太低，使用队列信号的应用程序可能失败。

同时还应更改哪些其他可调参数的值?

ksi_send_max 用以限制每个进程队列信号的数量。由于 **ksi_alloc_max** 基于每个系统， **ksi_send_max** 基于每个进程，因此 **ksi_alloc_max** 可调参数应始终更大。

请注意， **ksi_send_max** 的缺省值为 **32** ， **ksi_alloc_max** 的缺省值为 '**nproc * 8**' 。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。未来的 HP-UX 版本可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

ksi_alloc_max 由 HP 开发。

另请参阅

kill(2)、 sigqueue(2)、 ksi_send_max(5)。

名称

ksi_send_max - 用以限制每个进程队列信号的数量

值

缺省

32

允许值

32 到 **maxint (0x7fffffff)**

说明

ksi_send_max 为对每个进程发送者可以提交及当前接收者未决的队列信号数量的限制。强制限制基于每个进程发送者。 *ksi* 代表“内核信号信息”并使用相关队列信号信息识别条目。每个队列信号一个。队列信号由 **sigqueue** 系统调用、计时器到期、POSIX 实时消息队列及异步 I/O 使用。用户生成的信号（通过 **kill** 和 **raise** ）不会排队。

应该由谁来更改此可调参数？

任何人。

对于更改的限制

对此可调参数的更改将在下次重新引导时生效。

何时应增加此可调参数的值？

频繁或大量使用那些使用队列信号的设备时，可能需要增加此值。当 **sigqueue** 系统调用返回 [EAGAIN] 时，应该增加此可调参数。

增加此值的副作用是什么？

内存使用增加，但仅在使用时增加。每个已分配的条目都是 96 字节。

何时应降低此可调参数的值？

仅要控制应用程序使用队列信号时。

降低此值的副作用是什么？

如果太低，使用队列信号的应用程序可能失败。

同时还应更改哪些其他可调参数的值？

ksi_alloc_max 用以限制系统级队列信号的数量。由于 **ksi_alloc_max** 是基于每个系统的， **ksi_send_max** 是基于每个进程的，因此 **ksi_alloc_max** 可调参数总是应当更大。

请注意， **ksi_send_max** 的缺省值为 **32** ， **ksi_alloc_max** 的缺省值为 '**nproc * 8**' 。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。未来的 HP-UX 版本可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

ksi_send_max 由 HP 开发。

另请参阅

kill(2)、 sigqueue(2)、 ksi_alloc_max(5)。

名称

lang - 支持的语言说明

说明

HP-UX NLS （本地语言支持）为处理和自定义多种语言的需求提供支持。要为特定的语言启用 NLS 支持， HP-UX 系统上必须存在一个语言定义。调用 **locale -a** 命令（请参阅 *locale*(1) ）显示了有关特定 HP-UX 系统当前支持的语言的信息。

HP-UX 缺省的处理语言为 **POSIX** 。 **POSIX** 提供一个不使用 NLS 功能进行处理的环境。该环境基于 7 位代码 USASCII 字符集。

POSIX 和 **C** 是等效的并且可以互换使用的。

作者

lang 由 HP 开发。

另请参阅

locale(1)、 localedef(1M)、 setlocale(3C)、 environ(5)。

名称

langinfo - 语言信息常量

概要

#include <langinfo.h>

说明

此头文件包含用于标识 **langinfo** 数据项目的常量（请参阅 *nl_langinfo*(3C)）。 *items* 模式在 **<nl_types.h>** 中给出。可以定义以下常量（类别指示以哪一种 **setlocale()** 类别定义每一个项目）。

常量	类别	说明
CODESET	**LC_CTYPE**	代码集名称，如 **iso88591** 和 **eucJP** 。
D_T_FMT	**LC_TIME**	确定 **date** 、 **getdate()** 和 **strftime()** 的 **%c** （日期和时间）指令格式的字符串。
D_FMT	**LC_TIME**	用于确定 **date** 、 **getdate()** 和 **strftime()** 的 **%x** （日期）指令格式的字符串。
T_FMT	**LC_TIME**	用于确定 **date** 、 **getdate()** 和 **strftime()** 的 **%X** （时间）指令格式的字符串。
T_FMT_AMPM	**LC_TIME**	以具有 **AM_STR** 和 **PM_STR** 的 12 小时制格式表示的时间形式。
AM_STR	**LC_TIME**	使用 12 小时制时间格式的上午字符串（英文表示为 **AM** ）。
PM_STR	**LC_TIME**	使用 12 小时制时间格式的下午字符串（英文表示为 **PM** ）。
DAY_1	**LC_TIME**	一周中第一天的名称（英文表示为 **Sunday** ）。
...	**...**	...
DAY_7	**LC_TIME**	一周中第七天的名称。
ABDAY_1	**LC_TIME**	一周中第一天的缩写名称（英文表示为 **Sun** ）。
...	**...**	...
ABDAY_7	**LC_TIME**	一周中第七天的缩写名称。
MON_1	**LC_TIME**	阳历年中第一个月的名称。
...	**...**	...
MON_12	**LC_TIME**	第十二个月的名称。
ABMON_1	**LC_TIME**	第一个月的缩写名称。
...	**...**	...
ABMON_12	**LC_TIME**	第十二个月的缩写名称。

ERA	**LC_TIME**	纪元说明段，它描述如何在一个语言环境中计数并显示每一个纪元中的年。每一个纪元说明段的格式如下： *direction***:***offset***:***start_date***:** *end_date***:***era_name***:***era_format* 根据下文中的描述。将有足够多必需的纪元说明段来描述各不同的纪元。各纪元说明段由分号分隔。 注意，一个纪元的开头可能不是该纪元中最早的点，它可能是最晚的点。例如，基督纪元 BC 以 AD 1 年 1 月 1 日的前一天开始，年数增加表示时间更早。 *direction*： **+** 或 **-** 字符。 **+** 字符指示距 **start_date** 更近的年份数字小于那些距 **end_date** 更近的年份数字。 *offset*： 该纪元中距 *start_date* 最近的年份数字。 *start_date* ： 以格式 *yyyy***/***mm***/***dd*, 表示的日期，其中 *yyyy* 、 *mm*、"*dd"* 分别是纪元开始的年、月和日的数字。 AD 1 之前的各个年份以负数表示。 *end_date* ： 纪元的结尾日期，以与 *start_date* 相同的格式表示，或者是以两个特殊值 **-*** 或 **+*** 中的一个表示。值 **-*** 指示结尾日期是时间的开始。值 **+*** 指示结尾日期是时间的结尾。 *era_name*： 纪元，与 **%EC** 转换规范相对应。 *era_format*： 纪元中年的格式，与 **%EY** 转换规范相对应。
ERA_D_FMT	**LC_TIME**	没有为纪元指定单独的纪元格式时，用于确定 **date** 和 **strftime()** 的 **%E**（帝号/纪元号和年份）指令格式的字符串（请参阅 *localedef*(1M)）。
ERA_D_T_FMT	**LC_TIME**	语言环境中合适的替代日期和时间格式，与 **%Ec** 字段描述符相对应。
ERA_T_FMT	**LC_TIME**	语言环境中合适的替代日期和时间格式，与 **%EX** 字段描述符相对应。
ALT_DIGITS	**LC_NUMERIC**	数字的替代符号，与 **%O** 转换规范修饰符相对应。该值包含由分号分隔的字符串。第一个字符串是与 0 相一致的替代符号，第二个字符串是与 1 相一致的替代符号，等等。最多可以指定 100 个替代符号字符串。
RADIXCHAR	**LC_NUMERIC**	基数字符（英文表示为“decimal point”）。返回的字符串与 **localeconv()** 返回的结构中的 **decimal_point** 元素相同。
THOUSEP	**LC_NUMERIC**	千位分隔符。返回的字符串与 **localeconv()** 返回的结构中的 **thousands_sep** 元素相同。

YESEXPR **LC_MESSAGES** 肯定响应扩展正则表达式。

NOEXPR **LC_MESSAGES** 否定响应扩展正则表达式。

YESSTR **LC_MESSAGES** 对是/否问题的肯定响应（过时：使用 **YESEXPR** ）。

NOSTR **LC_MESSAGES** 对是/否问题的否定响应（过时：使用 **NOEXPR** ）。

CRNCYSTR **LC_MONETARY** 对于货币符号，如果后面跟有数字，则前面为 **-** ；如果跟在数字之后，则前面为 **+** ；如果替换基数，则前面为 **.** 。例如， **-DM** 将用于 **de_DE.iso88591** （ **DM1234,56** ）， **+ Kr** 用于 **da_DK.iso88591** （ **1234,56 Kr** ），而 **.$** 用于 **pt_PT.iso88591** (**1234$56**)。有关替代货币格式信息，请参阅 *localeconv*(3C) 。

DIRECTION **LC_CTYPE** 指示文本方向的值。当前定义的值包括 **null** 、 **0** 和 **1** 。 **null** 值或 **0** 值指示一行中的字符按从左到右排列，而各行按从顶部到底部排列。 **1** 值指示一行中的字符按从右向左排列，而各行按从顶部到底部排列（此常量是一个属于 HP 所有的项目，可能会更改，且可能无法移植到其他平台上）。

CONTEXT **LC_CTYPE** 指示字符上下文分析的字符串。字符串 **null** 或 **0** 指示不需要上下文分析。字符串 **1** 指示需要阿拉伯上下文分析。

ALT_DIGIT **LC_NUMERIC** 映射到 ASCII 对等字符串的一串字符 **0123456789*b*+-.,eE** （其中 *b* 是一个空格）。这也是对输出的反向映射。不假定数字的字符代码值是相邻的，或它们是单字节值。字符串是一个空值，指示该语言没有替代数字（此常量是一个属于 HP 所有的项目，可能会更改，且可能无法移植到其他平台上）。

ALT_PUNCT **LC_CTYPE** 映射到 ASCII 对等字符串的一串字符 ***b*!"#$%&'()*+,-./:;<=>?@[\]^_‘{|}˜** （其中 *b* 是一个空格）的美国用法。这也是对输出的反向映射。不假定标点符号的字符代码值是相邻的，或它们是单字节值。如果任何一个标点符号都没有对等的替代符号，那么就在替代标点字符串中使用 ASCII 代码。字符串是一个空值，指示该语言没有替代标点符号（此常量是一个属于 HP 所有的项目，可能会更改，且可能无法移植到其他平台上）。

YEAR_UNIT **LC_TIME** 年份符号。经常需要使用此符号来指定亚洲语言的日期（此常量是一个属于 HP 所有的项目，可能会更改，且可能无法移植到其他平台上）。

MON_UNIT **LC_TIME** 月份符号（此常量是一个属于 HP 所有的项目，可能会更改，且可能无法移植到其他平台上）。

DAY_UNIT	**LC_TIME**	天的符号（此常量是一个属于 HP 所有的项目，可能会更改，且可能无法移植到其他平台上）。
HOUR_UNIT	**LC_TIME**	小时符号。经常需要使用此符号来指定亚洲语言的时间（此常量是一个属于 HP 所有的项目，可能会更改，且可能无法移植到其他平台上）。
MIN_UNIT	**LC_TIME**	分钟符号（此常量是一个属于 HP 所有的项目，可能会更改，且可能无法移植到其他平台上）。
SEC_UNIT	**LC_TIME**	秒的符号（此常量是一个属于 HP 所有的项目，可能会更改，且可能无法移植到其他平台上）。
CHARMAP	**LC_COLLATE** **LC_CTYPE**	用于编译此语言环境的字符映射名称（此常量是一个属于 HP 所有的项目，可能会更改，且可能无法移植到其他平台上）。

警告

建议使用 **strftime()** 来访问在 **category** 中定义的日期和时间（请参阅 *strftime*(3C)），使用 **LC_TIME** 和 **localeconv()** 来访问 **RADIXCHAR**、 **THOUSEP** 和 **CRNCYSTR** 的相应信息（请参阅 *localeconv*(3C)）。

作者

langinfo 由 HP 开发。

另请参阅

date(1)、 localedef(1M)、 getdate(3C)、 localeconv(3C)、 nl_langinfo(3C)、 setlocale(3C)、 strftime(3C)、 lang(5)。

名称

lcpu_attr - 动态启用或禁用缺省处理器集的 LCPU 属性

值

保证安全

0 （禁用）

缺省值

0 （禁用）

禁用缺省处理器集的 LCPU 属性

允许值

0 （禁用）或 **1** （启用）

说明

此可调参数可以动态启用 (**1**) 或禁用 (**0**) 缺省处理器集中的逻辑处理器 (LCPU) 属性。在支持超线程技术的系统上，每个超线程均以一个 LCPU 表示。

如果启用 LCPU 属性，则会对缺省处理器中的处理器核心启用超线程。如果禁用 LCPU 属性，则缺省处理器集中的物理处理器的行为如同单线程处理器核心。

有关管理除缺省处理器集之外的其他处理器集中的 LCPU 属性的信息，请参阅 *pset_setattr*(2) 。

更改此可调参数的人员

希望更改缺省处理集中的超线程功能的系统管理员。

更改限制

对于没有超线程功能的平台，或者在固件级别禁用了超线程的系统，此可调参数不起作用。在支持超线程的系统上，必须启用固件功能，以动态启用（或禁用）缺省处理器集中的 LCPU 属性。

应该何时启用此可调参数选项

要启用缺省处理集中的 LCPU 属性，应打开此可调参数，以便利用超线程。如果要使系统上运行的应用程序体现出性能优势，应保持启用 LCPU 属性。

启用此可调参数的负面影响

如果启用 LCPU 属性，某些工作负荷可能会遇到性能下降的问题。

应该何时禁用此可调参数选项

某些应用程序可能会遇到性能下降的问题，或者体现不出使用超线程的任何性能优势。在此种情况下，应关闭此可调参数。

禁用此可调参数的负面影响

在关闭 LCPU 的情况下，应用程序将无法利用超线程。

应该同时更改的其他可调参数

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

lcpu_attr 由 HP 开发。

另请参阅

psrset(1M)、 pset_assign(2)、 pset_bind(2)、 pset_create(2)、 pset_ctl(2)、 pset_destroy(2)、 privgrp(4)。

名称

ldapux - LDAP-UX 概述

说明

LDAP-UX 是一种新的服务，它允许管理员使用 LDAP 目录服务器来存储有关用户和组的信息，及其他系统信息。

本联机帮助页提供有关 LDAP-UX 的常规信息：可用的名称服务、 LDAP-UX 不支持的功能以及如何获得有关 LDAP 的详细文档资料。

LDAP-UX 服务

LDAP-UX 提供了四种服务： **NSS_LDAP** 、 **PAM_LDAP** 、 **PAM_AUTHZ** 和 **NIS/LDAP Gateway** 。

- **NSS_LDAP**

  名称服务交换模块，使用 LDAP 从一个目录服务器检索系统信息，该目录服务器使用 *getpwent*(3C) 、 *getgrent*(3C) 、 *gethostent*(3N) 、 *getrpcent*(3C) 、 *getservent*(3N) 、 *getprotoent*(3N) 、 *getnetent*(3N) 和 *getnetgrent*(3C) 调用系列。该模块在配置 **nsswitch.conf** 文件（请参阅 *nsswitch.conf*(4) ）时将使用名称 **ldap** 。LDAP-UX 产品附带了一个名为 **/etc/nsswitch.ldap** 的 **nsswitch.conf** 示例文件。有关详细信息，请参阅 *nsswitch.conf*(4) 。

- **PAM_LDAP**

  一个 *PAM*(3) 模块，可利用 LDAP 目录服务器提供的身份验证工具。该模块在 **/etc/pam.conf** 文件中配置（请参阅 *pam.conf*(4) ）。LDAP-UX 产品附带了一个名为 **/etc/pam.ldap** 的 **pam.conf** 示例文件。有关详细信息，请参阅 *pam_ldap*(5) 。

- **PAM_AUTHZ**

  通过 PAM 的 **pam_authz** 服务模块提供的功能，管理员可以根据 **/etc/passwd** 文件中找到的 **netgroup** 信息或在访问策略文件 **/etc/opt/ldapux/pam_authz.policy** 中定义的访问规则，来控制登录到系统的用户。有关详细信息，请参阅 *pam_authz*(5)。

- **NIS/LDAP Gateway**

  可将 NIS 请求转换为 LDAP 请求的服务。有关详细信息，请参阅 *ypldapd*(8) 。

不支持的功能

尽管 **NSS_LDAP** 和 **PAM_LDAP** 支持用户数据和组数据的大多数用法，但是，使用 LDAP-UX 时不支持下列命令：

chsh	LDAP-UX 不支持。
chfn	LDAP-UX 不支持。
passwd	只有 **PAM_LDAP** 支持，而 **NSS_LDAP** 不支持。

LDAP-UX 文档

LDAP-UX 的 **readme** 文件中提供的文档介绍了 LDAP-UX 配置，通过使用 **setup** 程序（位于 **/opt/ldapux/config** 目录中）可完成此配置。另请参阅 /opt/ldapux/README 。

有关详细信息，请参阅 **http://docs.hp.com/hpux/internet** 上的《Installing and Administering LDAP-UX Client Services》 (J4269-90006) 和《LDAP-UX Client Services Release Notes》。有关订购信息，请参阅 *manuals*(5) 联机帮助页。

文件

/etc/nsswitch.conf	nsswitch 的配置文件
/etc/nsswitch.ldap	使用 LDAP 的 nsswitch 的示例配置文件
/etc/pam.conf	PAM 的配置文件
/etc/pam.ldap	使用 **pam_ldap** 的 PAM 的示例配置文件
/opt/ldapux/config	包含 LDAP-UX 配置工具的目录

另请参阅

ldapentry(1)、 ldapclientd(1M)、 getpwent(3C)、 getgrent(3C)、 pam(3)、 ldapclientd.conf(4)、 nsswitch.conf(4)、 pam_ldap(5)、 ypldapd(8)。

位于 /opt/ldapux/README 的产品文档资料。

位于 **http://docs.hp.com/hpux/internet** 的 LDAP 联机手册：
《Installing and Administering LDAP-UX Client Services》 (J4269-90006)
《LDAP-UX Client Services Release Notes》

名称

libcrash - 崩溃转储访问库

概要

```
#include <libcrash.h>

int     cr_open(const char *path, CRASH **cb, int flags);
int     cr_verify(CRASH *crash_cb, int flags);
cr_info_t *cr_info(CRASH *crash_cb);
int     cr_uncompress(CRASH *crash_cb, const char *pathname,
              uint64_t size, int flags);
int     cr_isaddr(CRASH *crash_cb, uint64_t pagenum, int *avail);
int     cr_read(CRASH *crash_cb, void *buf, uint64_t mem_page,
            int *num_pages);
int     cr_set_node(CRASH *crash_cb, int node_num,
             int *old_node_num);
void    cr_perror(CRASH *crash_cb, int error);
int     cr_close(CRASH *crash_cb);
```

说明

libcrash 是一个库，它提供对系统崩溃转储的访问。通过此库来访问一个转储与此崩溃转储的格式无关（下文中描述了几种格式）。它也与转储的位置无关，此转储可位于原始转储设备中、文件系统的文件中，或两种情况的混合。通过使用 **/dev/mem** 驱动程序，也可将一个正在运行的系统的内存当作“转储”来处理。

所有通过此库来对转储进行的访问，都以调用 **cr_open()** 开始。从此调用中返回的崩溃转储描述符是所有其他 **libcrash** 调用的一个必要参数。它们是：

cr_verify() 通过检查弥补转储的所有文件的大小与校验和来验证转储的完整性。

cr_info() 返回一个指向一个结构的指针，此结构包含关于此转储以及生成此转储的计算机和内核的信息。

cr_uncompress() 在崩溃转储中准备一个供使用的文件，方法是解压缩此文件（如果需要），并验证文件的大小与校验和。此函数由库在内部使用，用于访问物理内存映像，它可由调用方使用来访问内核和内核模块文件。

cr_isaddr() 给出关于一个特定的物理内存页在转储的计算机上是否有效的信息，如果有效，则此页的内容是否包括在此转储中。

cr_read() 从转储中读取页。

cr_set_node() 设置由 **cr_read()** 和 **cr_isaddr()** 使用的节点号，从而访问此节点私有内存区域中的内存。

cr_perror() 将与另一个库调用所返回的错误或警告代码相对应的错误或警告信息输出到标准错误中。

cr_close() 终止对崩溃转储的访问，释放所有由此库所分配的空间。

上述每一个调用都有它们自己的手册页，其中更详细地描述了它们的用法。

崩溃转储格式

有四种当前的系统崩溃转储格式:

COREFILE (V0) 此格式供 HP-UX 10.01 以前的版本使用，它由一个包含物理内存映像的单个文件组成，且在文件地址偏移量和内存地址之间存在 1 对 1 的对应关系。通常会有一个包含内核映像的相关文件。

COREDIR (V1) 此格式在 HP-UX 10.10、10.20 和 10.30 中使用，它由一个 **core.***n* 目录组成，此目录包含一个 **INDEX** 文件、内核 (vmunix) 文件和大量包含物理内存映像部分的 **core.***n***.***m* 文件。

CRASHDIR (V2) 此格式在 HP-UX 11.00 和以后的版本中使用，它由一个 **crash.***n* 目录组成，此目录包含一个 **INDEX** 文件、内核和所有动态加载的内核模块文件，以及大量的 **image.***m***.***p* 文件，其中，这些文件中的每一个文件都包含物理内存映像和描述哪些内存页已转储而哪些没有转储的元数据部分。

PARDIR (V5) 此格式在 HP-UX 版本 11i 1.0 及以后的版本中使用。在结构上这与 **CRASHDIR** 格式相似，因为它由一个 **crash.***n* 目录组成，此目录包含一个 **INDEX** 文件、内核和所有动态加载的内核模块文件，以及大量的 **image.***m***.***p* 文件，其中，这些文件中的每一个文件都包含物理内存映像和描述哪些内存页已转储而那些没有转储的元数据部分。除了主要的 **INDEX** 文件以外，还有辅助的索引文件，它们包含一些元数据，这些元数据描述包含内存页的映像文件。此格式将在转储设备已压缩内存映像的时候使用。请参阅 *crashconf*(1M) 。

将来可能会加入其他格式，例如磁带归档格式。

高速缓存

实现缓存机制是为了提高分析 **PARDIR** 格式崩溃转储时的性能。此缓存机制用于保留未压缩的页，这样，缓存中的页可满足后续请求的要求。缺省情况下，此将禁用缓存机制。通过设置环境变量 **LIBCRASH_ENABLE_CACHE**. 可启用此机制。这将在崩溃转储目录中创建缓存的文件。

返回值

libcrash 中大多数的调用返回一个整数状态值。返回值为零，指示成功。尽管请求的操作已完成，返回值为正数指示某种警告。返回值为负数，指示发生某种错误，从而阻止请求操作的完成。库返回的值有:

CR_OK 成功。

CRWARN_NOEXPECTED 由于没有记录崩溃转储中一个或多个文件的期望大小或校验和，因此无法验证此转储的完整性。转储可能会损坏。

CRWARN_NOACTUAL 由于无法计算崩溃转储中一个或多个文件的校验和，所以无法对照期望值检查此校验和。转储可能会损坏。

CRWARN_SWAPPEDON 包含一部分崩溃转储的原始设备已被交换，因此此转储很可能已损坏。

CRWARN_MISMATCH 崩溃转储中一个或多个文件的大小或校验和未匹配期望的值。此转储可能已损坏。

CRERR_NOPAGE 发出了一个对在目标计算机上不存在的内存地址的读取或写入请求。

CRERR_READONLY 发出了一个对崩溃转储的写入请求。通过 **/dev/mem** 驱动程序，仅支持对正在运行的系统进行写入操作。

CRERR_WRONGDUMP 期望包含部分转储的原始转储设备却不包含。它可能已被交换活动或更近的转储覆盖。

CRERR_WRONGHOST 一部分崩溃转储仍驻留于被转储的系统而非当前系统的一个转储设备上。

CRERR_NONODE 指定的节点号不存在。

CRERR_ERRNO 出现系统错误。有关特定错误，请查询 **errno** 。请注意， **errno** 的某些值在库的相关环境中具有特定的含义。它们包括：

[ENOEXEC] 不能解压缩一部分崩溃转储。

[ENOTDIR] 指定的转储路径名既不是一个纯文本文件，也不是一个包含 INDEX 文件的目录，也不是 **/dev/mem** 伪驱动程序。

**** **errno** 的其他值具有它们的传统含义。

作者

libcrash 由 HP 开发。

另请参阅

cr_close(3)、cr_info(3)、cr_isaddr(3)、cr_open(3)、 cr_perror(3)、cr_read(3)、cr_set_node(3)、 cr_uncompress(3)、cr_verify(3)。

名称

libcres - libc 的函数子集

说明

libcres 库可以提高链接到 **libc** 的共享库版本的应用程序的性能。与此库中包含的函数位于 **libc** 的共享版本时相比，经常调用此库中包含的这些函数的应用程序可以更有效地执行。这一改进完全取决于应用程序对这些函数的使用情况。

libcres 库和应用程序可以通过使用 **ld** 中所述的链接程序选项 **-lcres** 来进行链接（请参阅 " *ld*(1)）。

下面列出了 **libcres** 中的一些接口:

> **abs()**、**div()**、**ffs()**、**labs()**、**ldiv()**、**llabs()**、**lldiv()**、**memchr()**、
> **memcmp()**、**memmove()**、**memset()**、**strcat()**、**strchr()**、**strcmp()**、
> **strcpy()**、**strcspn()**、**strlen()**、**strncat()**、**strncmp()**、**strncpy()**、
> **strrchr()**、**strstr()**、**swab()**

此库内容在未来版本中会进行更改。

警告

如果开发人员将某个应用程序和 **-lcres**相链接，则不能使用用户定义的具有相同名称的函数来代替此库中包含的函数。开发人员必须禁止在该应用程序中使用这些名称，否则将可能产生不可预想的结果。

另请参阅

ld (1)。

http://docs.hp.com 上的《HP-UX Linker and Libraries User’s Guide》。

名称

limits - 特定于实现方法的常量

概要

#include <limits.h>

说明

下列各符号在 **<limits.h>** 中定义，并在此手册的描述性文本中使用。以 **HP-UX** 值开头的列会列出应用程序编写者为了所有 HP-UX 系统间的可移植性而应采用的值。

值后面的符号解释如下:

- **+** 在某些 HP-UX 系统中，实际的限制可能会大于指定的值。
- **++** 在某些 HP-UX 系统上，实际的限制可能会大于 **<limits.h>** 中指定的值。
- **-** 在某些 HP-UX 系统中，实际的限制可能会小于指定的值。
- **=** 实际的限制总是等于指定的值，且在各个 HP-UX 系统中都相同。
- ***** 仅当预处理器宏 **_XPG2** 已由编译标记 **-D_XPG2** 定义，或由源文件中的 **#define** 指令在源文件包括 **<limits.h>** 之前定义时，才定义此限制的名称。
- **#** 为此限制定义的值可能不是一个编译时常量。在运行时，定义的值始终为一个整数表达式。

某些限制会随系统配置的不同而不同，并可使用 *sysconf*(2) 进行动态确定。某些限制根据文件系统或与特定文件相关联的设备而有所不同，并可使用 *pathconf*(2) 来确定。其他限制已过时，因为它们连同其他限制是冗余的，或在可移植的应用程序中没有用。提供这些限制仅仅是为了实现其他系统对应用程序的可输入性，从而支持那些遵循《X/Open Portability Guide》，第二期的应用程序，以及对 HP-UX 以前版本的向后兼容。 **_XPG2** 标记不应在新的应用程序中定义。

通过在编译中包括 **<limits.h>** 文件，应用程序可测试合适的限制，从而确定其可在特定的系统中操作，还是甚至可能会更改其行为以便匹配此系统，从而提高其在限制设置和系统的一个变化范围内的可移植性。

常量	说明	**HP-UX** 值
ARG_MAX	*exec*(2) 参数的最大长度，以字节为单位，包括环境数据	***5120 +****
CHAR_BIT	**char** 中的位数	***8 =***
CHAR_MAX	**char** 的最大整数值	***127 =***
CHAR_MIN	**char** 的最小整数值	***-128 =***
CHILD_MAX	每一个用户 ID 的同步进程最大数	***25 +-****
CLK_TCK	每秒钟的时钟周期数	***50 +#***
DBL_DIG	**double** 的精度数位	***15 +***
DBL_MAX	**double** 的最大正值	***1.7976931348623157e+308 +***

DBL_MIN	**double** 的最小正值	***4.9406564584124654e-324 -***
FCHR_MAX	以字节为单位的最大文件地址偏移量	***INT_MAX +-****
FLT_DIG	**float** 的精度数位	***6 +***
FLT_MAX	**float** 的最大正值	***3.40282346638528860e+38 +***
FLT_MIN	**float** 的最小正值	***1.40129846432481707e-45 -***
INT_MAX	**int** 的最大十进制值	***2147483647 +***
INT_MIN	**int** 的最小十进制值	***-2147483648 -***
LINE_MAX	单个行中的最大字符数	***2048 =***
LINK_MAX	对单个文件的最大链接数	***32 767 +****
LOCK_MAX	系统锁表中的最大条目数	***32 +-****
LONG_BIT	**long** 中的位数	***32 +***
LONG_MAX	**long** 的最大十进制值	***2147483647 +***
LONG_MIN	**long** 的最小十进制值	***-2147483648 -***
MAX_CANON	终端规范输入行中的最大字节数	***512 +****
MAX_CHAR	终端输入队列中的最大字节数	***MAX_INPUT =****
MAX_INPUT	终端输入队列中的最大字节数	***512 +****
NAME_MAX	路径名部分中的最大字节数	***14 +****
NL_ARGMAX	对 NLS *printf*(3S) 和 *scanf*(3S) 函数的调用中的最大“数字”值	***9 =***
NL_MSGMAX	NLS 消息清单中的最大消息数	***32767 +***
NL_SETMAX	NLS 消息清单中的最大集合数	***255 +***
NL_TEXTMAX	NLS 消息字符串中的最大字节数	***8192 +***
NGROUPS_MAX	每个进程的最大补充组数	***20 ++***
OPEN_MAX	一个进程中可打开的最大文件数	***60 +****
PASS_MAX	口令中的最大字符数	***8 +***
PATH_MAX	路径名称中的最大字符数（排除空结束符）	***1023 +****
PID_MAX	一个进程 ID 的最大值	***30000 +***
PIPE_BUF	写入一个管道的最大原子字节数	***8192 +****
PIPE_MAX	在一次写入操作中可写入一个管道的最大字节数	***INT_MAX +***
PROC_MAX	系统中的最大同步进程数	***84 +-****
SCHAR_MAX	**signed char** 的最大整数值	***127 =***
SCHAR_MIN	**signed char** 的最小整数值	***-128 =***
SHRT_MAX	**short** 的最大十进制值	***32767 +***
SHRT_MIN	**short** 的最小十进制值	***-32768 -***
STD_BLK	一个物理 I/O 块中的字节数	***512 +***
SYSPID_MAX	各系统进程的最大进程 ID	***4 +-****
SYS_NMLN	*uname*(2) 返回的字符串长度	***8 +****

SYS_OPEN	在系统中打开的最大文件数	***120 +-****
TMP_MAX	由 *tmpnam*(3S) 生成的唯一名称的最大数	***17576 +***
UCHAR_MAX	**unsigned char** 的最大整数值	***255 =***
UID_MAX	对于一个用户或组 ID 来说最小的难以得到的值	***2147483647 +***
UINT_MAX	**unsigned int** 的最大十进制值	***4294967295 +***
ULONG_MAX	**unsigned long** 的最大十进制值	***4294967295 +***
USHRT_MAX	**unsigned short** 的最大十进制值	***65535 +***
USI_MAX	**unsigned int** 的最大十进制值	***UINT_MAX =****
WORD_BIT	一个“字”中的位数 **(int)**	***32 +***

举例

UID_MAX 具有一个值为 **2147483647 +** 的 HP-UX 值，它意味着在所有的 HP-UX 系统中，对于一个用户或组 ID 来说，最小的难以得到的值至少是 2147483647。一个特定的系统可能会支持大于 2147483647 的用户或组 ID，在这种情况下， **<limits.h>** 文件将 **UID_MAX** 设置为一个更高的值；然而，任意一个采用这样一个更高的值的应用程序不保证可移植到所有 HP-UX 系统上。

NGROUPS_MAX 包含一个值为 **20 ++** 的 HP-UX 值，这意味着在所有 HP-UX 系统上，每个进程的最大补充组数至少为 20。某特定的系统可能能够支持每个进程超过 20 个补充组，在这种情况下，**sysconf(_SC_NGROUPS_MAX)** 将返回一个比 **<limits.h>** 中指定的值更大的值；但是，不能保证任意一个采用这样更高值的应用程序，能够移植到所有 HP-UX 系统上。

作者

limits 由 HP 开发。

另请参阅

exec(2)、 fcntl(2)、 fork(2)、 getgroups(2)、 link(2)、 lockf(2)、 open(2)、 pathconf(2)、 sysconf(2)、 uname(2)、 write(2)、 printf(3S)、 scanf(3S)、 tmpnam(3S)、 passwd(4)、 values(5)、 termio(7)。

符合的标准

<limits.h>：AES、SVID3、XPG2、XPG3、XPG4、FIPS 151-2、POSIX.1、POSIX.2、ANSI C

名称

livedump - 为进行调试将操作系统状态保存到文件系统的功能。

说明

使用 **Live Dump** 功能，可以将正在运行的操作系统的状态保存到文件系统，而不需要停止（或重新引导）系统。保存的转储可用于调试。

Live Dump 以系统崩溃转储访问库 **libcrash** 可以读取的一种现有转储格式保存信息。请参阅 *libcrash*(5) 。

Live Dump 将映像和相关的文件保存在目录 *path*/**ldump.***n* 中。目录名中尾随的 *n* 是一个数字，每次使用相同的 **path** 运行 **Live Dump** 时，该数字都会加 1。该数字存储在文件 *path*/**lbounds** 中。如果配置的路径中不存在 **lbounds** 文件，则创建该文件，并将缺省值 2 存储在该文件中。

Live Dump 支持阻塞和非阻塞调用。在阻塞调用中，仅当完成 **Live Dump**

后，才将控制权返回给调用方。

如果使用非阻塞调用，则将控制权返回给被调用方，并且，用户可以使用 **livedump** 命令查询 **Live Dump** 的状态。

对于阻塞模式调用，将在用户控制台上输出 **Live Dump** 消息；对于非阻塞模式调用，将在 syslog 文件 **/var/adm/syslog/syslog.log** 中存储 **Live Dump** 消息。

Live Dump 支持使用页分类的选择性转储，这种转储类似于崩溃转储。 **Live Dump** 使用文件系统空间保存操作系统状态。用户可以配置保持可用状态的文件系统空间量。 **Live Dump** 保存的操作系统状态采用映像文件格式。用户也可以配置每个映像的最大大小。用户可以设置 **Live Dump** 在文件系统上保存系统状态的位置。

可以使用 **livedump** 实用程序查询（或更改） **Live Dump** 配置。有关详细信息，请参阅 *livedump*(1M) 。

Live Dump 保存的转储不是系统内存的快照，而是在一段时间内的状态。由于系统状态不断发生变化（ **Live Dump** 正在运行时），因此，获取的转储只是暂时性的。由于这种暂时性，转储中的某些数据结构可能会受到破坏。

定义

minfree 完成 **Live Dump** 后，保持可用状态的空间量。如果 **Live Dump** 检测到可用磁盘空间低于 *minfree* 值，则会停止保存转储。在这种情况下，保存的转储将不可调试。

chunksize 单个物理内存映像文件的大小。有关详细信息，请参阅 *livedump*(1M) 。

pageclass 系统内存类。有关定义的系统内存类的详细信息，请参阅 *crashconf*(2) 。 **Live Dump** 不支持保存缓冲区缓存和未用页。

sid 与每个 **Live Dump** 会话关联的会话 ID。 **Live Dump** 针对每次调用生成此 ID，以便唯一地标识 **Live Dump** 会话。用户可以使用此 ID 指定要异常中止的 **Live Dump** 会话。

path 用于保存 **Live Dump** 的目录。如果不指定此选项，则将 **/var/adm/crash** 用作保存转储的缺省目录。

作者

此处介绍的 **Live Dump** 功能由 HP 开发。

文件

/var/adm/syslog/syslog.log 由 **Live Dump** 保存的消息

另请参阅

livedump(1M)、 crashconf(2)、libcrash(5)。

名称

man - 用于设置联机帮助页格式的宏

概要

man *file* ...

nroff **-man** [*option*]... [*file*]...

说明

man 宏由 **man** 和 **nroff** 命令使用（参阅 *man*(1) 和 *nroff*(1)）— 也可由 **troff** 使用（参阅第三方文档）— 用来格式化《HP-UX 参考手册》及其他相关参考手册中的联机版本的帮助页。 **man** 命令将调用 **nroff** 。

man 和 nroff 缺省设置

缺省页面大小为 66 行，每行 85 个字符（8.5×11 英寸），文本区域为 60 行，每行 75 个字符。连字符关闭，段落为左对齐，右侧未对齐。

troff 缺省设置

缺省页面大小为 8.5×11 英寸，文本区域为 6.5×10 英寸。打印大小为 10 点，垂直行间距为 12 点。连字符打开，段落左右均对齐。

其他缺省设置

在处理诸如 **.I** 、 **.RB** 和 **.SM** 等字体和大小设置宏之后，以及在每个段落之前，打印字体和大小重置为缺省值。制表位只能由宏 **.DT** 和 **.TH** 使用和设置，任何其他宏都不能使用或设置制表位。 **.TH** 宏将调用 **.DT** （参阅下面的内容）。

选项

可以为 **nroff** 或 **troff** 指定以下选项。但它们不能用于 **man** 命令。

-rs1 减小 **troff** 的尺寸：页面大小减为 6×9 英寸，文本区域减为 4.75×8.375 英寸，打印大小减为 9 点，垂直行间距减为 10 点。 **nroff** 会忽略此选项。

-rV2 将特定参数设置为适合于特定的 Versatec 打印机的值：行宽为 82 个字符 (ens)；页面宽度为 84 行；禁用下划线。注意不要将此选项与 **man** 命令的 **-Tvp** 选项相混淆，后者在某些操作系统上可用，但在 HP-UX 上不可用。

宏、字符串和数值汇总

这里对所定义的 **man** 宏、字符串和数值进行了汇总，并在随后的各小节中加以详细说明。

.B 将 *text* 设为粗体。
.BC 将 *word* 交替设置为粗体和等宽。
.BI 将 *word* 交替设置为粗体和斜体。
.BR 将 *word* 交替设置为粗体和罗马体。
.C 将 *text* 设为等宽。
.CB 将 *word* 交替设置为等宽和粗体。
.CD 标识命令名。

.CI	将 *word* 交替设置为等宽和斜体。
.CR	将 *word* 交替设置为等宽和罗马体。
.CT	标识引用标题。
.DT	恢复缺省制表符设置。
.EM	标识强调。
.ER	标识错误名。
.EV	标识环境变量名。
.GT	标识词汇表术语。
.HP	段落开始时悬挂缩进。
.I	将 *text* 设为斜体。
.IB	将 *word* 交替设置为斜体和粗体。
.IC	将 *word* 交替设置为斜体和等宽。
.IP	用可选标记开始缩进的段落。
.IR	将 *word* 交替设置为斜体和罗马体。
.KC	标识按键。
.P	开始正常段落。
.PD	设置段落间距。
.PM	使用 Bell System 专有的子页脚。
.PP	开始正常段落。
.RB	将 *word* 交替设置为罗马体和粗体。
.RC	将 *word* 交替设置为罗马体和等宽。
.RE	结束相对边距缩进。
.RI	将 *word* 交替设置为罗马体和斜体。
.RS	开始相对边距缩进。
.RV	标识返回值。
.S3	插入第三级标题。
.SC	标识系统常量名。
.SH	插入章节标题。
.SM	减小一个点来打印 *text* 。
.SS	插入子章节标题。
.TH	开始新的联机帮助页并定义页眉和页脚。
.TP	开始有标记的段落。
***R**	注册商标。
***S**	更改为缺省打印大小。
***(Tm**	商标。
\n(IN	左侧文本边距；缺省边距缩进和段落缩进。
\n(LL	行长度，包括 **\n(IN** 。
\n(PD	段落间距。

宏参数

所有宏参数都具有位置性，并且可以省略（从右侧开始）。每个参数都是一个 *word* ，如下所述。

mi 边距增量。这是左侧文本边距的增量。如果将其省略，则缺省设置为 **\n(IN** 基本单位 (**u**) 。 *mi* 的缺省计量为 ens (**n**) 。左侧文本边距的基本值为 **\n(IN** 。

in 段落缩进。这是段落缩进行的缩进量。如果将其省略，则缺省设置为由以下最近的段落宏建立的值：由 **.HP** 、 **.IP** 或 **.TP** 显式或缺省建立的值；由 **.P** 、 **.PP** 、 **.RE** 或 **.RS** 隐式建立的值。 *in* 的缺省计量为 ens (**n**) 。

text 由零到六个 *word* 组成。如果 *text* 为空，则对下一个包含要打印的文本的行进行特别处理。例如，可以使用 **.I** 将整行变为斜体，或者使用 **.SM** 并后跟 **.B** 以生成小粗体文本。

word 一串由空格（而不是制表符）分隔的字符。使用引号 ("[*word*]") 可以在 *word* ("*string string*") 中包含空格或者指定一个空 *word* ("")。（不间断的非填充空格 (\ ) 不是分隔空格）。

标题宏

.TH *t1 s2 c3 n4 a5*

设置章节标题或条目标题。 *t1* 、 *s2* 、 *c3* 、 *n4* 和 *a5* 都是 *word* 。

t1 条目标题。

s2 章节编号。 *t1* 与 *s2* 组合放在括号中，构成了页面标题的左上角和右上角。

c3 额外的注释，例如“需要的可选软件”。它放在底部一行的中间，占据两、三行页面标题空间。

n4 其他表示法，例如“仅适用于 300（或 400）系列”。它位于第一个页面标题行上的标题与章节号之间的中间位置。

a5 支持备用名，例如对应于在 *t1* 中指定的 C 函数名的 FORTRAN 例行程序名。

所获得的输出的形式如下：

t1(*s2*)	*n4*	*t1*(*s2*)
a5		*a5*
	c3	

.SH *text* 在此处放置章节标题 *text* ，例如概要。章节标题从左边距处开始。由于它们通常均为大写，因此它们在 **troff** 中以小一个点数的大小打印。

.SS *text* 在此处放置子章节标题 *text* ，例如 **Options** 。子章节标题始于左边距与正常文本缩进之间。

.S3 *text* 在此处放置第三级标题 *text* ，例如子标题。第三级标题在正常文本缩进处开始。

段落宏

.P

.PP 开始一个块段落。将 *in* 重置为 **\n(IN** ，“取消”由以前的 **.HP** 、 **.IP** 和 **.TP** 宏设置的任何值。

.HP *in* 开始一个带悬挂缩进的段落。文本在当前边距处开始。第二个及后续输出行缩进。

.TP *in* 用悬挂标记开始一个缩进的段落。下一个包含要打印的文本的输入行将视作标记。标记在当前边距处开始。如果标记能放在缩进内，则段落文本将在同一行上的缩进处开始。如果标记不能放在缩进内，则段落文本将在下一行上的缩进处开始。

.IP *t in* 与带有标记 *t* 的 **.TP** *in* 相同。通常用来获得一个不带标记的缩进段落。

.RS *mi* 将当前左边距增加 *mi* 。如果省略了 *mi* ，则缺省设置为 *in* 的当前值。针对新的边距级别，将段落缩进 *in* 设置为 **\n(IN** 。最多可以指定九个 **.RS** 增量级别。可以通过 **.RE** 宏撤消边距增量，并通过 **.TH**、 **.SH**、 **.SS** 和 **.S3** 标题宏将其重置为基本边距。

.RE *k* 返回到第 *k* 个左边距设置（初始时， *k*=1 ； *k*=0 相当于 *k*=1 ）。如果省略了 *k* ，则返回到前一个边距值。段落缩进 *in* 恢复到在执行对应的 **.RS** 宏之前时的相应值。

.PD *pd* 将段落间距设置为 *pd* 垂直间距。如果省略了 *pd* ，则将段落间距设置为缺省值：在 **nroff** 中为 1 行。在 **troff** 中为 0.4 行。 *pd* 的计量为垂直行间距 (**v**) 。

字体宏

.B *text* 将 *text* 设为粗体。

.C *text* 将 *text* 设为等宽字体。请参阅“警告”部分。

.I *text* 将 *text* 设为斜体。

（这里没有 **.R** （罗马体）宏，但可以在需要时使用某个 **.***XY* 组合）。

.*XY a b* 将字体 *X* 中的 *a* 与字体 *Y* 中的 *b* 连接在一起，这两种字体最多可以变换出六个 *word* 。字体符号 *X* 和 *Y* 可以是 **B** （粗体）、 **C** （等宽）、 **I** （斜体）和 **R** （罗马体），其组合如下：

	.CB	**.IB**	**.RB**
.BC		**.IC**	**.RC**
.BI	**.CI**		**.RI**
.BR	**.CR**	**.IR**	

有关等宽字体的更多信息，请参阅“警告”部分。

.SM *text* 使 *text* 比缺省点大小小一个点数。这在 **nroff** 中不起作用。

特殊宏

这些宏用来标识联机帮助页中的通用文本元素。它们有助于在 HP 样式中提供一致的字体用法，并改进了向其他格式系统的转换。

以适当的字体或格式设置第一个参数。第二个参数 *punctuation* 被设置罗马体，提供给所连接的标点符号。这两个参数是由字体宏连接起来的。

.CD *commandname punctuation*

commandname 是一个命令名，通常在第 1 章或 1M 联机帮助页中定义，例如 **man** 。它以等宽字体显示。

.CT *citationtitle punctuation*

citationtitle 是某个文档的名称，例如《HP-UX 参考手册》。它以斜体显示。（联机帮助页参考使用标准的 **.IR** 宏）。

.EM *emphasis punctuation*

emphasis 是要强调的一个字或词组，例如， 不要。请不要过多地使用强调。它以斜体显示。（变量名使用标准的 **.I** 宏）。

.ER *errorname punctuation*

errorname 是对应于由某个函数分配给 **errno** 的值的错误名，在联机帮助页的“错误”部分中进行了说明。它以罗马体显示，并括在方括号中。例如， **.ER EIO .** 将显示为 [EIO] 。

.EV *environvarname punctuation*

environvarname 是某个环境变量名，例如 **PATH** 。它以等宽字体显示。

.GT *glossterm punctuation*

glossterm 是某个词汇表术语，或是在联机帮助页中定义的供日后使用的术语，例如 “路径名”。它以粗体显示。

.KC *keycap punctuation*

keycap 是某个键盘按键的名称，例如 **Tab** 。它以粗体显示。

.RV *returnvalue punctuation*

returnvalue 是某个函数的数字返回值或某个命令的退出状态，通常在“返回值”或“退出状态”部分中定义。它通常用于数字表达式，例如 **0** 、 **>3** 以及 **<>0** ，但不用于文字说明，例如“非零”。它以等宽字体显示。

.B *systemconstant punctuation*

systemconstant 是某个操作系统常量名，例如 **PATH_MAX** 。它以等宽字体显示。请参阅 *getconf*(1) 。

其他宏

.DT 恢复缺省制表符设置：在 **nroff** 中为每 5 个 ens。在 **troff** 中为每 3.6 个 ems。

.PM *sf* 生成 Bell System 专有的子页脚。

sf	生成
P	**PRIVATE**
N	**NOTICE**
BP	**BELL LABORATORIES PROPRIETARY**
BR	**BELL LABORATORIES RESTRICTED**

字符串

定义以下字符串引用：

***R** 注册商标符号：在 **nroff** 中显示为 **(Reg.)** ，在 **troff** 中使用 **\(rg** 内置宏。

***S** 更改为缺省打印大小。这是作为一个 **\s** 内置宏执行的。

***(Tm** 商标指示符 TM，显示为上标（如果可能的话）。

数值

定义有以下数值引用：

\n(IN 相对于章节标题的左侧文本边距缩进以及 *mi* 和 *in* 的缺省值：在 **nroff** 中为 5 个 ens。在 **troff** 中为 3.6 个 ems。 **IN** 以基本单位 (**u**) 来表示。

\n(LL 行长度包括 **\n(IN** ： 在 **nroff** 中为 75 个字符。在 **troff** 中为 6.5 英寸。另请参阅“选项”部分。 **LL** 以基本单位 (**u**) 来表示。

\n(PD 当前段落间距。由 **.PD** 设置。 **PD** 以垂直行间距 (**v**) 来表示。

计量

nroff 和 **troff** 使用若干标尺指示符来实现适当的水平和垂直计量。许多宏参数都具有缺省的计量单位。因为所有对数字变量的赋值都会转换成基本单位 (**u**) ，因此在赋值和引用值时要非常小心，这一点很重要。

标尺		基本单位	
指示符	计量	**nroff**	**troff**
c	厘米	240/2.54	*D*/2.54
i	英寸	240	*D*
m	em	*C*	**p***S*
n	en = em/2	*C*	**p***S*/2
p	点 = 1/72 英寸	240/72	*D*/72
P	pica = 1/6 英寸	240/6	*D*/6
u	基本单位	1	1
v	垂直行间距	*V*	*V*

C 输出设备的字符宽度。

D 输出设备的每英寸点数 (dpi)。

S 以点计的当前打印大小。

V 以基本单位计的当前垂直行间距。

字体约定

《 *HP-UX* 参考手册》中的条目使用以下字体约定：

罗马体 常规字体。

italic 用于变量以及表示某个参数的其他词，此参数可能会获取一个用户定义的值或变量值。也用于 *emphasis* 。

粗体 主要用于标题，偶尔也用于首次出现的术语或要进行定义的术语。

等宽字体	用于所有这样的文字，即，用作程序等中的键盘命令或命令行选项时，按照其实际显示打印的文字。

页脚

在页脚中使用的字符串是由 **.TH** 宏初始化的。 **)H** 缺省设置为一个非打印（空）字符串。 **]W** 缺省设置为对联机帮助页进行格式设置的日期，例如 **Formatted: July 5, 1995** 。在 HP 联机帮助页中， **)H** 设置为公司名称“ Hewlett-Packard Company”； **]W** 设置为版本信息“HP-UX 10.10 发行版：1995 年 11 月”。

这种安排使用户及第三方软件供应商在创建他们自己的手册条目时，能够直接控制页脚行左右字段中的内容，根据需要显示公司名称、发行版本等等信息。 **)H** 显示在左侧， **]W** 显示在右侧，页码显示在中间。这些字符串可以在 **.TH** 宏调用之后的任意位置处定义，但要出现在第一页结束之前。例如，以下源文件段：

```
.TH man 5
.ds )H XYZ Company
.ds ]W Release 2.3: May 1996
```

将产生一个具有如下文本形式的页脚：

XYZ Company	– 1 –	Release 2.3:May 1996

表

tbl 预处理器（请参阅 *tbl*(1) ）可用来在联机帮助页中插入表。 **man** 宏允许使用标准的 **tbl** 宏： **.TS** 、 **.T&** 和 **.TE** 。它们不支持 **mm** 宏扩展 **.TS H** 和 **.TH** （参阅 *mm*(5) ）。

一般来讲，应避免在表内使用 **man** 宏，特别是字体宏，因为它们会产生无法预测的特殊结果。请使用 **nroff** 或 **troff** 的内置宏。例如，要指定粗体，请使用内置宏 **\fB** ，或者，更为常见的是使用 **\f3** 。要插入水平空白行，请使用 **.PP** 或 **.IP** 宏，视哪个在表的前面而定，但随后必须避免使用 **center** 和 **expand** 表格式规范。否则，缩进会出现混乱。

警告

对于印刷手册，HP 不再使用“名称”部分来准备目录和索引。相反，此信息将编码为每个联机帮助页源文件末尾的注释，可以根据需要通过各种工具和程序对其进行访问。“名称”部分仍用于 **whatis** 数据库，如下所述。

用于打印《 *HP-UX* 参考手册》的宏程序包在每个条目的概要部分增加了字间距（以便使之更加清楚）。

除了上面提到的宏、字符串和数值以外，还定义有许多内部宏、字符串和数值寄存器。其中包括：

- **nroff** 或 **troff** 处理器预定义的名称。
- 宏 **th** 。
- 数值寄存器 **:m** 。
- **)***x* 、 **]***x* 、和 **}***x* 形式的宏、字符串和数值名称，其中 *x* 代表某些字母数字字符。
- *XY* 形式的宏、字符串和数值名称，其中 *X* 和 *Y* 是大写字母。

字体

nroff 仅使用三种字体：罗马体、斜体和粗体，分别指定为字体位置 1、2 和 3。宏程序包中的等宽字体宏（ .C、 **.RC**、 **.IC**、 **.BC**、 **.CR**、 **.CI**、 **.CB** ）可用粗体字体模拟等宽字体，因为所有 **nroff** 输出都是等宽或打字机格式的。

要在 **troff** 中使用真正的等宽字体，请将每个等宽字体宏中对应的字体 3 规范更改为字体 4，并使用一个 **troff .fp** 请求在位置 4 处安装一个等宽字体，例如：

.fp 4 CW

whatis 数据库

每个联机帮助页的“名称”部分将由 **catman** （请参阅 *catman*(1M) ）处理并在 **whatis** 数据库中创建一个条目，供 **man** 命令的 **-f** 和 **-k** 选项使用。 **catman** 会将“名称”部分的各行处理成以下格式的单个行：

name[**,** *name*]... **-** *explanatory-text*

连字符（打印为 **-** 或 **\-** ）、en 短线 (**\(mi**) 和 em 短线 (**\(em**) 将以同等方式处理。行中最后一个空格-连字符-空格将成为名称和解释文本之间的分隔点。

文件

/usr/share/lib/macros/an

man 宏。

/usr/share/lib/tmac/tmac.an

由 **man** 调用。确定宏的源 (**.so**) 位于 **/usr/share/lib/macros/an** 中。可以在这里指定其他宏文件，以满足联机帮助页对于宏的其他要求。

/usr/share/lib/whatis

由来自联机帮助页“名称”部分的字符串构成的文件，由 **catman** 创建，并由 **man -k** 和 **-f** 选项使用。

另请参阅

col(1)、 man(1)、 neqn(1)、 nroff(1)、 tbl(1)、 catman(1M)、 mm(5)。

名称

manuals - 访问和订购 HP-UX 文档

说明

最新的 HP-UX 用户手册和白皮书可在 HP 技术文档资料网站 **http://www.docs.hp.com** 上获得。

要订购 Instant Information CD 上的手册，请访问 **http://docs.hp.com** ，然后单击 **Where to buy**。

要订购打印的手册，请访问 **http://software.hp.com** ，然后单击 **Product category** 列表中的 **Documentation** 。

名称

math - 数学函数、常量和类型

概要

#include <math.h>

说明

此文件包含了所有数学库中函数的声明（在 (3M) 一节中介绍）。

对于 HP Integrity 服务器，包含在 **-fpwidetypes** 选项下的编译中，此文件定义了类型

extended Integrity 服务器 80 位双精度扩展型。

quad 符合 IEEE 754 标准的 128 位浮点型。在 HP-UX 上， **quad** 与 **long double** 同义。

它定义了类型

float_t 浮点型，至少与 **float** 宽度相同。对于 PA-RISC， **float_t** 为 **float** 。对于 Integrity 服务器，如果 **FLT_EVAL_METHOD** 等于 0，则 **float_t** 为 **float** ；如果 **FLT_EVAL_METHOD** 等于 1，则 **float_t** 为 **double** ；如果 **FLT_EVAL_METHOD** 等于 -2，则 **float_t** 为 **extended** 。

double_t 浮点型，至少与 **double** 宽度相同。对于 PA-RISC， **double_t** 为 **double** 。对于 Integrity 服务器，如果 **FLT_EVAL_METHOD** 等于 0 或 1，则 **double_t** 为 **double** ；如果 **FLT_EVAL_METHOD** 等于 -2，则 **double_t** 为 **extended** 。

对于 Integrity 服务器，根据使用的编译选项 **-fpeval=float** （缺省）、 **-fpeval=double** 或 **-fpeval=extended** , **FLT_EVAL_METHOD** 的值为 0、1 或 -2。

它定义了下列常量，对于 Integrity 服务器，这些常量可能用于初始化静态和汇聚：

NAN 静态 NaN（非数字）值（ **float** 型）。

INFINITY 正无穷大（ **float** 型）。

HUGE_VAL 能以 **double** 型表示的最大值（ **double** 型）（IEEE 正无穷大）。

HUGE_VALF 能以 **float** 型表示的最大值（ **float** 型）（IEEE 正无穷大）。

HUGE_VALL 仅适用于 Integrity 服务器，能以 **long double** 型表示的最大值（ **long double** 型）（IEEE 正无穷大）。

HUGE_VALW 仅适用于 Integrity 服务器，能以 **extended** 型表示的最大值（ **extended** 型）（IEEE 正无穷大）。要使用 **-fpwidetypes** ，请使用 **HUGE_VALW** 选项进行编译。

HUGE_VALQ 仅适用于 Integrity 服务器，等效于 **HUGE_VALL** 。要使用 **-fpwidetypes** ，请使用 **HUGE_VALQ** 选项进行编译。

此文件定义了下列整型常量，这些整型常量即为 **ilogb()** 函数返回的特例值：

FP_ILOGB0 如果其参数为零，则由 **ilogb()** 返回。

FP_ILOGBNAN
如果其参数为 NaN，则由 **ilogb()** 返回。

对于 Integrity 服务器，它定义了下列整型常量，这些整型常量标识 **<math.h>** 函数支持的错误处理方法:

MATH_ERRNO
指明对 ISO/IEC C99 **errno** 规范的支持。

MATH_ERREXCEPT
指明对 ISO/IEC C99 异常标记规范的支持。

math_errhandling
如果使用了 **+Olibmerrno** 编译选项（非缺省），则定义为 **MATH_ERRNO** ，否则定义为 **MATH_ERREXCEPT** 。

为了方便用户，它定义了下列数学常量（double 型）：

M_E 自然对数的底 (*e*) 。

M_LOG2E 以 2 为底的 *e* 的对数。

M_LOG10E 以 10 为底的 *e* 的对数。

M_LN2 2 的自然对数。

M_LN10 10 的自然对数。

M_PI 圆周率。（也有几个 π 的分数、倒数及平方根： **M_PI_2** 、 **M_PI_4** 、 **M_1_PI** 、 **M_2_PI** 和 **M_2_SQRTPI**）。

M_SQRT2 2 的正平方根

M_SQRT1_2 1/2 的正平方根。

有关各种与计算机相关的常量的定义，请参阅 **<values.h>** 头文件的说明。

要使用类型 **extended** 或 **quad** 之中的一种，或者宏 **HUGE_VALW** 或 **HUGE_VALQ** 之一，请使用 **-fpwidetypes** 选项进行编译。

文件

/usr/include/math.h

另请参阅

intro(3)、 complex(5)、 fenv(5)、 values(5)。

符合的标准

<math.h>：SVID3、XPG4.2、ANSI C、ISO/IEC C99

名称

max_acct_file_size - 定义最大记账文件大小

值

无故障

2,560,000 bytes

缺省值

2,560,000 bytes

允许值

在 2,560,000 和 2,147,483,647 字节之间。

建议值

2,560,000 bytes

说明

max_acct_file_size 为限制最大记账文件大小的动态可调参数。请参阅 *acct(2)* 和 *acct(4)* 。如果 **max_acct_file_size** 设置的值不是记账记录大小的倍数，则除非达到了最大限制，否则此可调参数将以无提示方式四舍五入为下一个完整记录。在这种情况下，此可调参数将以无提示方式四舍五入为最近的完整记录中。

此规则的唯一例外是当 **max_acct_file_size** 设置回其缺省值时。系统将在没有四舍五入的情况下接受 2,560,000 , 即使它不是记账记录大小的倍数也是如此。如此以保留与先前 HP-UX 版本的二进制兼容性。

应该由谁来更改此可调参数?

希望允许记账文件增长到缺省值之上的系统管理员。

对于更改的限制

无。

何时应增加此可调参数的值?

如果需要比缺省值更大的记账文件时。

增加此值的副作用是什么?

使用大小不超过 2.5 MB 的记账文件的记账工具可能无法处理更大的文件。

何时应降低此可调参数的值?

缺省值也是最小值时。如果增加 **max_acct_file_size** 的值，在不需要大型记账文件的情况下，可以再次降低此值。

降低此值的副作用是什么?

无。

同时还应更改哪些其他可调参数的值?

无。

警告

所有 HP-UX 内核可调参数都是针对于特定版本的。未来的 HP-UX 版本可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

max_acct_file_size 由 HP 开发。

另请参阅

acct(2)、acct(4)、acctsuspend(5)、acctresume(5)。

名称

max_async_ports - 可随时打开的异步磁盘端口的最大数量

值

保证安全

1024

缺省值

4096

允许值

1 - 4194304

建议值

此可调参数没有建议值。同时访问驱动程序的进程数量可用作设置值的标准。

说明

异步磁盘驱动程序为裸磁盘提供执行高性能 I/O 的接口。进程不直接读取或写入裸磁盘，而是对异步驱动程序写入请求，然后异动驱动程序对磁盘驱动程序进行适当的调用。

每个打开异步磁盘驱动程序的进程都分配有一个端口。驱动程序使用此端口跟踪 I/O 和此进程的其他内部资源。打开异步磁盘驱动程序的数量受可用端口数量的限制。

更改此可调参数的人员

任何用户。

更改限制

如果允许使用新值，对此可调参数的变更则将立即生效；否则变更将在下次引导时生效。始终可以立即接受在许可的范围内增加此值的大小。减小此值可能会立即生效，也可能会延迟到下次引导时生效，这取决于当前正使用的端口的数量或者曾经使用的端口的最大数量。

应该何时增加此可调参数的值

当需要打开驱动程序的进程的数量大于可调参数的当前值时。

增加此可调参数值的负面影响

从内核内存分配端口资源。如果设置的值较高，会导致驱动程序所使用的内核内存稍有增加。那些可能需要内核内存的其他内核组件会受到影响。但是，系统为每个端口保留的内存量都很低。所以，仅当实际分配新的端口时，才会明显地占用大量的内存。

应该何时降低此可调参数的值

实际不必要降低此可调参数的值。但是，如果多个进程正在打开异步的驱动程序和分配端口，那么降低此值将可以限制可供分配的端口的最大数量，以及驱动程序可以分配的内存的最大数量。

降低此可调参数值的负面影响

同时打开的数量受此可调参数的限制。这可能影响需要具有更高数量访问驱动程序的进程的应用程序的性能。

应该同时更改的其他可调参数

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

max_async_ports 由 HP 开发。

名称

maxdsiz、maxdsiz_64bit - 任何用户进程的数据段的最大大小（以字节为单位）

值

缺省值

32 位： **1 GB**

64 位： **4 GB**

允许值

32 位最小值： **0x40000**

32 位最大值： **0xfffff000**

64 位最小值： **0x40000**

64 位最大值： **0x3ffbffff000**

说明

HP-UX 系统上的用户程序由五个不连续的虚拟内存段组成：文本（或代码）、数据、堆栈、共享的和 I/O。每个段都占用一段已设定大小上限的结构化定义的虚拟地址空间范围，但由于 **maxtsiz** 、 **maxdsiz** 和 **maxssiz** 可调参数的强制约束，文本、数据段和堆栈段的最大值可能会略小。

此可调参数定义了 32 位和 64 位进程的静态数据存储段的最大大小。数据存储段包含了固定的数据存储，例如使用 **sbrk()** 和 **malloc()** 在 main()、字符串和空间中分配的全局变量、数组变量、静态变量和局部变量。

更改此可调参数的人员

任何用户。

更改限制

对此可调参数的更改只会影响在此更改之后启动的进程。此外，如果某个进程修改了其数据段的 **rlimit** ，则修改后的限制将制约所有子进程，将来再对 **maxdsiz** 进行任何修改都不起作用。指定的值预计将是基页大小的倍数。有关详细信息，请参阅 *getconf(1)* 中对 **_SC_PAGE_SIZE** 的说明。如果指定的值不是基页大小的倍数，则将四舍五入为该基页大小的最接近的倍数。

应该何时增加此可调参数的值

如果用户进程接收到以下 [ENOMEM] 错误消息，则应增加此可调参数的值：

exec(2): data exceeds maxdsiz

或者

exec(2): data exceeds maxdsiz_64bit

这可能会导致进程失败，也可能不会，具体取决于程序代码。

增加此值的负面影响

通过定义增加此可调参数的值可以为每个进程提供更大的数据段。这意味着 **maxdsiz** 和 **maxssiz** 限制了每个进程所能保留或使用的交换空间的大小。因此，使用更多虚拟地址空间并不直接意味着能够使用更多的物理地址空间，因为虚拟页可能会被交换出来。

如果计算机上的交换空间已接近其容量，则增加此可调参数的值会增加每个进程的可保留交换空间的量。这将允许某个出现内存泄漏的进程，或使用大量内存的恶意程序保留太多的交换空间进程，从而导致系统上的交换空间耗尽。

应该何时降低此可调参数的值

如果计算机上的交换空间非常紧张，而那些占用大量交换空间的程序将会影响其他关键用户进程的运行，则应该降低此可调参数的值。

降低此值的负面影响

如果计算机上的交换空间已接近其容量，那么降低此可调参数的值会限制为每个进程保留的交换空间量，并且会导致需要使用大量交换空间的进程收到 [ENOMEM] 错误。

应该同时更改的其他可调参数值

应考虑 **maxssiz** 可调参数，因为它也限制了进程堆栈段的交换空间使用情况。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

maxdsiz 由 HP 开发。

另请参阅

getconf(1)、 getrlimit(2)、 setrlimit(2)、 maxtsiz(5)、 maxssiz(5)。

名称

maxfiles - 每个进程的文件描述符的初始（软）最大数量

值

保证安全

60

缺省值

缺省值在运行时计算，它依赖系统上的物理内存量。对于小型内存系统（小于 1 GB），缺省值为 512。对于内存超过 1 GB 字节的系统，缺省值为 2048，或 2K。

允许值

最小值为 32。最大值为 1048576，或 1M。此值将进一步限制为其必须等于或小于硬限制 **maxfiles_lim** 。

指定一个正整数值。

建议值

在运行时，如果将可调参数设置为大于 409600 或 400K 的值，将发出警告。这超出了已测试的限制。

在运行时，如果此值不是文件描述符组块大小的倍数（8 的倍数），将发出警告。这不是一个严重警告，只是提供给管理员的一个消息。

说明

maxfiles 指定了对于在任何给定的时间打开文件时允许进程拥有的文件描述符的初始缺省数。进程可能增加其软限制并因此打开多于 **maxfiles** 的文件。

非超级用户进程可能在它们达到硬限制 **maxfiles_lim** 之前使用 **setrlimit()** 或 **ulimit()** 增加其软限制。

更改此可调参数的人员

此可调参数的值很少需要修改。然而，在运行使用大量的文件描述符的应用程序的系统上或在需要减少内存消耗的系统上时，可能需要修改此可调参数。

更改限制

此可调参数为静态的。为了使对此可调参数值的更改发生作用，需要重新引导系统。

应该何时增加此可调参数的值

在运行使用大量的文件描述符的应用程序的系统中，如果应用程序还没有增加其进程软限制，例如使用 **setrlimit()** ，可能需要增加此可调参数。

增加此值的负面影响

初始进程内存占用量由于每个进程的文件表更大而增长。

应该何时降低此可调参数的值

应降低此值，以便限制系统上每个进程的初始文件描述符的数目，从而减少内存消耗。

降低此值的负面影响

初始进程内存占用量由于每个进程的文件表更小而缩减。

应该同时更改的其他可调参数值

分配到 **maxfiles** 中的值必须小于或等于 **maxfiles_lim** 的值。内核在可调参数设置期间检查以确保这一点。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

maxfiles 由 HP 开发。

另请参阅

kctune(1M)、 sam(1M)、 gettune(2)、 settune(2)、 ulimit(2)、 setrlimit(2)、 maxfiles_lim(5)。

名称

maxfiles_lim - 每个进程的文件描述符的最大数目硬限制。

值

保证安全

1024

缺省值

4096

允许值

最小允许值为 32。最大允许值为 1048576 或 1M。此值将进一步限制为必须等于或大于软限制 **maxfiles** 。

指定一个正整数值。

建议值

在运行时，如果将可调参数设置为大于 409600 或 400K 的值，将发出警告。这超出了已测试的限制。

在运行时，如果此值不是文件描述符组块大小的倍数（8 的倍数），将发出警告。这不是一个严重警告，只是提供给管理员的一个消息。

说明

maxfiles_lim 指定了在任何给定的时间，某个进程针对打开的文件所允许具有的文件描述符数目的系统硬限制。这样，非超级用户进程便有可能增加其软限制，直至达到此硬限制。

更改此可调参数的人员

要使用大量文件描述符来运行应用程序的任何用户。

更改限制

maxfiles_lim 可调参数是动态的（所做的调整将在运行系统上立即生效）。

动态更改会影响系统中的所有现有进程，但以下进程除外：

- 分配的文件描述符数多于此新限制所允许的数目的进程，
- 通过调用 **setrlimit()** 或 **ulimit()** 专门设置了其限制的进程。

应该何时增加此可调参数的值

当多个进程需要打开大量文件描述符时，应增加此可调参数的值。

增加此值的负面影响

增加 **maxfiles_lim** 的值不会立即显现出效果。但是，这将允许任何进程分配更多的文件描述符，从而会潜在消耗更多系统内存。

应该何时降低此可调参数的值

应降低此值，以便限制系统上每个进程的打开文件描述符的数目并减少系统内存消耗时。

降低此值的负面影响

如果进程的文件描述符尚未超过新限制，则降低 **maxfiles_lim** 的值会限制其内存消耗。

应该同时更改的其他可调参数值

分配给 **maxfiles** 的值必须小于或等于 **maxfiles_lim** 的值。

可调参数设置过程中的内核检查将确保符合这些限制。

警告

对于大于 65535 的值，会影响与已过时的 **pstat_getfile()** 的兼容性。要确保操作正确，所有应用程序都必须使用 **pstat_getfile2()** ，而不能使用 **pstat_getfile()** 。如果系统仍包含使用 **pstat_getfile()** 的应用程序，则将 **maxfiles_lim** 设置为小于或等于 65535 可以保持兼容性，除非所查询的进程已经使用了 **setrlimit()** 将其打开文件的最大限制更改为大于 65535 的值。

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

maxfiles_lim 由 HP 开发。

另请参阅

kctune(1M)、 sam(1M)、 setrlimit(2)、 gettune(2)、 settune(2)、 pstat(2)、 ulimit(2)、 maxfiles(5)。

名称

max_mem_window - 用户可配置的组专用 32 位共享内存的最大数量

值

缺省值

0

允许值

最小值：**0** 个共享内存窗口

最大值：**65535** 个共享内存窗口

说明

PA-RISC 32 位体系结构中的进程通常可为诸如 I/O 映射、共享库、共享映射文件等共享全局象限 3 和 4。然而，用户可能希望仅用一组选定的进程以更有限的共享方式使用第 3 象限。内存窗口允许此功能。

如果将此可调参数设置为 **0**，那么对于共享内存 32 位程序始终使用全局 Q3 和 Q4。如果此可调参数大于用对此组的 Q3 私有定义的组中的 **0**，进程，那么如果 Q3 已满则共享全局 Q4。

更改此可调参数的人员

任何用户。

更改限制

对此可调参数的更改立即生效。

应该何时增加此可调参数的值

如果系统用户需要更多的组专用 Q3 区域，则应该增加此可调参数。每个处于活动使用状态的内存窗口都需要大约 1Mb 内核内存。当窗口不再使用时，其可用的内存将由系统收回。

增加此值的负面影响

内核钟需要使用更多的内存。

应该何时降低此可调参数的值

如果没有使用进程组，或此可调参数设置的数量集没有使用，则使此值大于实际需要的值没有任何意义。

降低此值的负面影响

内核使用更少的内存。如果将值设置为 **0**，则不允许使用任何组专用 Q3。

应该同时更改的其他可调参数值

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在

用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

max_mem_window 由 HP 开发。

另请参阅

getmemwindow(1M)、 setmemwindow(1M)、 services.window(4)。

名称

maxrsessiz, maxrsessiz_64bit - 对在基于 Itanium 的平台上的任意用户进程的 RSE 堆栈的最大大小（以字节为单位）

值

缺省值

32 位缺省： **0x800000 (8MB)**

64 位缺省： **0x800000 (8MB)**

允许值

32 位最小值： **0x40000**

32 位最大值： **0x17F00000**

64 位最小值： **0x40000**

64 位最大值： **0x40000000**

说明

基于 Itanium 的系统使用主内存中进程级的寄存器堆栈（有关详细信息，请参阅 Intel IA-64 体系结构软件开发人员手册，第 2 卷，第 6 章）。此堆栈在物理寄存器和主内存之间移动寄存器且由寄存器堆栈工程 (RSE) 进行维护。 **maxrsessiz** 确定此堆栈的大小。

应该由谁来更改此可调参数？

任何人。

对于更改的限制

对此可调参数的更改仅在下次重新引导时生效。

何时应增加此可调参数的值？

如果用户进程由于 RSE 堆栈溢出终止并出现 [SIGBUS] 错误，则应该增加 **maxrsessiz** 。

增加此值的副作用是什么？

用户进程将对 RSE 堆栈提供更多虚拟内存。这将导致数据分配可用的虚拟内存减少。

何时应降低此可调参数的值？

仅当如果系统上的交换空间非常紧张时降低此可调参数，这是因为对 RSE 堆栈的虚拟内存仍然需要匹配交换保留或分配。如果将 **maxrsessiz** 降低至缺省值之下，应该进行认真考虑，因为这可能导致在 RSE 堆栈溢出上的用户应用程序意外失败。

在没有完全具体了解计算机工作负荷的 RSE 堆栈使用的情况下，此可调参数最好保持不变。

降低此值的副作用是什么？

降低此可调参数将限制对每个进程使用 RSE 堆栈的可用内存量。这可能导致使用大型 RSE 堆栈要求的进程终止，并返回 [SIGBUS] 错误。

同时还应更改哪些其他可调参数的值？

无。

警告

所有 HP-UX 内核可调参数都是针对于特定版本的。未来的 HP-UX 版本可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

maxrsessiz 由 HP 开发。

另请参阅

maxssiz(5)、 maxdsiz(5)、 maxtsiz(5)。

名称

maxssiz、maxssiz_64bit - 任何用户进程的堆栈的最大大小（以字节为单位）

值

缺省值

32 位：**0x800000** (**8MB**)

64 位：**0x10000000** (**256MB**)

允许值

32 位最小值：**0x40000**

32 位最大值：**0x17F00000**

64 位最小值：**0x40000**

64 位最大值：**0x80000000**

说明

HP-UX 系统上的用户程序由五个不连续的虚拟内存段组成：文本（或代码）、数据、堆栈、共享的和 I/O。每个段都占用一段已设定大小上限的结构化定义的虚拟地址空间范围。但由于 **maxtsiz** 、 **maxdsiz** 和 **maxssiz** 可调参数的强制约束，文本段、数据段和堆栈段的最大值可能会略小。

maxssiz 和 **maxssiz_64bit** 定义了 32 位和 64 位进程的堆栈段的最大大小。堆栈段包含了某个进程或线程相关环境切换上的寄存器的实际程序堆栈和存储空间。

更改此可调参数的人员

任何用户。

更改限制

对此可调参数的更改只会影响在此更改之后启动的进程。此外，修改了其堆栈段的 **rlimit** 的进程会将修改后的限制扩展到所有子进程，从而防止其将来对 **maxssiz**进行的任何修改。指定的值预计将是基页大小的倍数。有关详细信息，请参阅 *getconf*(1) 中对 **_SC_PAGE_SIZE** 的说明。如果指定的值不是基页大小的倍数，则将四舍五入为基页大小的最接近的倍数。

应该何时增加此可调参数的值

如果用户进程生成以下控制台错误消息，则应增加 **maxssiz** 的值：

Warning: maxssiz value too small

生成此错误消息的进程可能会因段违规错误 [SIGSEGV] 而终止并转储核心。

增加此值的负面影响

通过定义增加此可调参数的值可以为每个进程提供更大的堆栈段。这意味着 **maxdsiz** 和 **maxssiz** 将发挥限制作用，限制每个进程能够保留或使用的交换空间的量。因此，使用更多虚拟地址空间并不直接意味着使用更多物理

地址空间，因为虚拟页可能会被交换出。

如果计算机上的交换空间已接近其容量，则增加此可调参数的值会增加每个进程的可保留交换空间的量。这将允许某个出现内存泄漏的进程，或使用大量内存的恶意程序保留太多的交换空间进程，从而导致系统上的交换空间耗尽。

此外还要注意一点，对于 32 位用户进程，数据和堆栈是彼此相邻的。增加为堆栈段保留的虚拟地址空间的量将意味着要减少数据段的虚拟地址空间的量。也就是说，增加 **maxssiz** 可能会导致之前使用了所有（或几乎所有）可用数据区域的用户进程无法进行分配，并出现 [ENOMEM] 错误，即使此进程将 **maxdsiz** 设置为大于分配给数据的当前内存容量时也是如此。

在基于 Itanium®- 的系统的值、增加 **maxssiz_64bit** 的值将相应增加用于表示较大堆栈空间的内核数据结构。这可能会占用大量的额外交换空间，从而造成因缺少可保留的交换空间而使性能下降，或使应用程序无法正常运行的现象。

降低影响的一种方法是使用启动需要大量堆栈大小的应用程序的脚本。此脚本将增加 **maxssiz_64bit** 的值，启动应用程序，然后再将 **maxssiz_64bit** 降低至其原来的值。这样一来，特定的应用程序便可以利用增加了的堆栈空间，但又不会导致不需要增加堆栈的应用程序出现额外的堆栈增长。

应该何时降低此可调参数的值

如果计算机上的交换空间非常紧张，而那些占用大量交换空间的程序将会影响其他关键用户进程的运行，则应该降低此可调参数的值。

降低此值的负面影响

降低此可调参数的值将限制每个进程的堆栈所能使用的内存量。这可能会导致具有较大堆栈需求的进程可能会因 [SIGSEGV] 错误而终止。

应该同时更改的其他可调参数值

应考虑 **maxdsiz** 可调参数，因为它也可以限制进程数据段对交换空间的使用。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

maxssiz 由 HP 开发。

另请参阅

getconf(1)、 maxdsiz(5)、 maxtsiz(5)。

名称

max_thread_proc - 定义允许的最大并发线程数量每个进程。

值

无故障

256

缺省值

256

允许值

在 64 和 nkthread 之间

建议值

256

说明

max_thread_proc 为限制系统上每个进程允许的最大线程数量的动态可调参数。当调整 **max_thread_proc** 时，每个进程允许的最大线程数量将为 **max_thread_proc** 的新值。没有进程能创建使其新线程总数超过 **max_thread_proc** 。

应该由谁来更改此可调参数?

运行其系统每个进程都需要大量线程的应用程序的系统管理员。

对于更改的限制

在现有进程上调整 **max_thread_proc** 的作用未定义。但是，现有线程将不受影响。

何时应增加此可调参数的值?

当需要运行要求每个进程线程数量高于用当前设置 **max_thread_proc** 来满足的数量的应用程序时，应该增加 **max_thread_proc** 的值。如果创建线程失败，且 **EAGAIN** 的值为 **errno** ，则可能表示已经达到 **max_thread_proc**。但是，达到 **max_thread_proc** 不是创建线程失败并返回 **EAGAIN** 的唯一原因。如果已经达到 **nkthread** 系统可调参数或系统内存已经耗尽，可能返回 **EAGAIN** 。

增加此值的副作用是什么?

一组进程可能耗尽系统上系统级限制的线程。

何时应降低此可调参数的值?

如果有必要限制每个进程允许的线程数量时。

降低此值的副作用是什么?

某些需要大量线程的应用程序的行为可能与原来不同或者操作失败。

同时还应更改哪些其他可调参数的值?

可能需要查看 **nkthread** 的值。

警告

所有 HP-UX 内核可调参数都是针对于特定版本的。未来的 HP-UX 版本可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

max_thread_proc 由 HP 开发。

另请参阅

nkthread(5)。

名称

maxtsiz, maxtsiz_64bit - 任一用户进程的文本段的最大大小（以字节为单位）

值

缺省值

32 位：**96MB**

64 位：**1GB**

允许值

32 位最小值：**0x40000**

32 位最大值：**0x40000000**

64 位最小值：**0x40000**

64 位最大值：**0x3FFFFFFFFFF**

说明

HP-UX 系统上的用户程序由虚拟内存的五个不连续段组成：文本（或代码）、数据、堆栈、共享及 I/O。每个段都占用一段已设定大小上限的结构化定义的虚拟地址空间范围，但由于 **maxtsiz** 、 **maxdsiz** 和 **maxssiz** 可调参数的强制约束，文本段、数据段和堆栈段的最大值可能略小。

maxtsiz 控制文本段的大小，它是可被执行同一程序的多个进程共享的进程的只读可执行对象代码。例如，系统上的所有 vi 副本都使用相同的文本段。

更改此可调参数的人员

任何用户。

更改限制

对此可调参数的更改立即生效。其大小为基页大小的倍数。有关详细信息，请参阅 *getconf(1)* 中对 **_SC_PAGE_SIZE** 说明。如果指定的值不是基页大小的倍数，则将四舍五入为基页大小的最接近的倍数。

应该何时增加此可调参数的值

当用户进程接收带有下列消息的 [ENOMEM] 错误时应该增加 **maxtsiz** ：

exec(2): text exceeds maxtsiz

或者

exec(2): text exceeds maxtsiz_64bit

增加此值的负面影响

无。

应该何时降低此可调参数的值

尽管在系统性能方面没有理由这么做，但是仍然应该降低此可调参数的值，以限制正在运行的进程的文本大小。

降低此值的负面影响

无。

应该同时更改的其他可调参数值

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

maxtsiz 由 HP 开发。

另请参阅

getconf(1)、 maxdsiz(5)、 maxssiz(5)。

名称

maxuprc - 限制每个用户的并发用户进程的最大数量

值

无故障

256

缺省值

256

允许值

在 3 和（nproc -5 ）之间。

建议值

256

说明

maxuprc 是限制每个用户进程的最大数的动态的可调参数。只有超级用户才可以拥有超过 **maxuprc** 所限制的进程数量。

应该由谁来更改此可调参数？

系统管理员可以取决于系统使用情况更改 **maxuprc** 的值。

对于更改的限制

无。此可调参数为动态的。

何时应增加此可调参数的值？

如果用户需要比当前 **maxuprc** 所允许的更多的进程，则应该更改 **maxuprc** 的值。如果 **fork()** 失败并显示错误值 [EAGAIN] ，则指示此特定用户达到 **maxuprc** 。然而，这不是可能导致 **fork()** 失败并出现错误值 [EAGAIN] 的唯一原因。如果已经达到 **nproc** 系统可调参数或已耗尽系统内存，可能返回 [EAGAIN] 。

增加此值的副作用是什么？

增加 **maxuprc** 的值允许单个用户消耗更多的系统资源。

何时应降低此可调参数的值？

当个人用户运行太多的并发进程而独占系统资源时应该降低 **maxuprc** 的值。

降低此值的副作用是什么？

依赖于大量进程的应用程序行为可能不同或失败。现有的进程会继续运行，但导致用户超过 **maxuprc** 的新进程创建将会失败。

同时还应更改哪些其他可调参数的值？

当调整 **maxuprc** 时应该牢记 **nproc** 的值。

警告

所有 HP-UX 内核可调参数都是针对于特定版本的。未来的 HP-UX 版本可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

maxuprc 由 HP 开发。

另请参阅

nproc(5)。

名称

maxvgs - 可在系统上创建（或激活）的 LVM 卷组的最大数（已过时）

说明

maxvgs 可调参数已过时，并已被删除。

此可调参数指定了系统上可以被创建或激活的最大 LVM 卷组数量。当前在不需要此可调参数的情况下， LVM 可在系统上支持 256 个卷组；这等效于将已过时的 **maxvgs** 可调参数设置为 256。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。对于 HP-UX 11i v3 及此后的版本，此参数已过时。

作者

maxvgs 由 HP 开发。

另请参阅

lvm(7)。

名称

mesg - 引导时启用或禁用 System V IPC 消息（已过时）

说明

mesg 可调参数已过时。始终启用 System V IPC 消息子系统。

概述

System V 消息只是通过消息队列在协作进程之间传递的字节序列。消息可以为达到可调参数最大值的任意长度。大于 64 字节的消息（非可调参数）存储在保留的内核内存中。等于或小于 64 字节的消息则分配其消息标头，因此不消耗额外的内核结构来维持内存。每个消息都以一个应用程序特定号“键入”。每个消息队列都通过一个唯一的 ID 来引用，且可以包含多个单独的消息。

接收消息的进程可以获取“第一条”消息 (FIFO)，第一次指定的类型、第一组类型或等待显示指定类型。

警告

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

mesg 由 AT&T 开发。

另请参阅

ipcrm(1)、 ipcs(1)、 msgctl(2)、 msgget(2)、 msgrcv(2)、 msgsnd(2)、 msgmnb(5)、 msgmni(5)、 msgmbs(5)、msgtql(5)。

名称

mknod.h - 处理设备号的宏的头文件

概要

#include <sys/mknod.h>

说明

头文件 **<sys/mknod.h>** 定义了创建并解释设备标识号的宏，与 **mknod()** 系统调用系统调用一起使用（请参阅 *mknod*(2)）。

这些宏的使用是与结构相关的。有关如何选择主设备及从设备号的信息，请参阅系统的《System Administration Manual》。

mknod.h 包含宏

dev_t makedev(int major, int minor)

将主要部分和次要部分压缩为适于 **mknod()** 的 *dev* 参数的设备标识号，及以下两个宏：

int major(dev_t dev)
int minor(dev_t dev)

它将分别从设备标识号 *dev* 中提取主编号部分及次编号部分。

宏 **MINOR_FORMAT** 为一份 **printf()** 规范（请参阅 *printf*(3S)），以最适合特定的实现的格式打印从编号；它由 **ls** 命令的 long 格式（请参阅 *ls*(1)）用于显示设备文件的从编号。

编号的基数以 C 程序语言中相同的方式指定：没有十进制前导零、八进制前导零和十六进制前导 **0x** 。

另请参阅

ls(1)、 mknod(1M)、 mknod(2)、 printf(3S)。

名称

mm - 用于格式化文档的 MM 宏程序包

概要

mm [options] [files]

nroff -mm [options] [files]

说明

此程序包为各种各样的文档提供了格式化功能。文档输入及编辑的方式实质上不依赖于文档最终是在终端被格式化或是设置为照片类型。有关进一步的详细信息，请参阅以下参考资料。 **-mm** 选项使得 *nroff*(1) 和 *troff* 使用非压缩版本的宏程序包。

文件

/usr/share/lib/macros/mmn	非压缩形式的程序包
/usr/share/lib/tmac/tmac.m	非压缩形式的程序包指针

另请参阅

mm(1)、nroff(1)。

名称

mman - 内存映射定义

概要

#include <sys/mman.h>

说明

The **<sys/mman.h>** 头文件定义了和 **madvise()** 函数一起使用的下列符号常量:

MADV_NORMAL	不进行进一步的特殊处理。
MADV_RANDOM	期望随机页引用。
MADV_SEQUENTIAL	期望顺序页引用。
MADV_WILLNEED	将需要这些页。
MADV_DONTNEED	将不需要这些页。
MADV_SPACEAVAIL	确保保留资源。

定义下列符号常量以便与 **mmap()** 和 **mprotect()** 函数一起使用:

PROT_READ	可以读取区域。
PROT_WRITE	可以写入区域。
PROT_EXEC	可以执行区域。
PROT_NONE	不可以访问区域。

定义下列符号常量以便与 **mmap()** 函数一起使用:

MAP_FILE	映射一个文件。
MAP_ANONYMOUS	映射一个未命名内存区域。
MAP_VARIABLE	在实现计算地址放置区域。
MAP_FIXED	在指定地址放置区域。
MAP_SHARED	共享对映射区域进行的更改。
MAP_PRIVATE	对映射区域的更改为进程专用。

定义下列符号常量以便与 **msync()** 函数一起使用:

MS_SYNC	执行同步写入。
MS_ASYNC	执行异步写入。
MS_INVALIDATE	使缓存页面无效。

定义下列符号常量以便与 **msem_init()** 、 **msem_lock()** 和 **msem_unlock()** 函数一起使用:

MSEM_LOCKED	在锁定状态下创建信号量。
MSEM_UNLOCKED	在未锁定状态下创建信号量。
MSEM_IF_NOWAIT	如果信号量被锁定,则不等待。
MSEM_IF_WAITERS	如果信号量没有等待者,则不解除锁定。

定义 **typedef struct msemaphore** 以便与 **msem_init()** 、 **msem_lock()** 、 **msem_unlock()** 和 **msem_remove()** 函数

一起使用。

另请参阅

mmap(2)、 munmap(2)、 mprotect(2)、 msync(2)、 madvise(2)、 msem_init(2)、 msem_remove(2)、 msem_lock(2)、 msem_unlock(2)。

名称

msgmap - System V IPC 消息空间资源映射中的条目数量（已过时）

说明

msgmap 可调参数已过时并被删除。

此可调参数指定消息空间资源映射的大小（即所包含条目的数量），系统使用该资源映射来跟踪共享 IPC 消息空间中的空闲空间。但现在，内核中已更改消息空间分配机制，因此不再需要此可调参数。

有关 System V 消息队列的详细信息，请参考 *mesg*(5) 联机帮助页中的“概述”一节。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。对于 HP-UX 11i v3 及此后的版本，此参数已过时。

作者

msgmap 由 AT&T 开发。

另请参阅

msgrcv(2)、 mesg(5)、 msgmbs(5)、 msgmax(5)、 msgmnb(5)、 msgmni(5)、 msgseg(5)、 msgssz(5)、 msgtql(5)。

名称

msgmax - System V IPC 消息最大字节数（已过时）

说明

msgmax 可调参数已过时并被删除。

此可调参数指定了 System V 消息队列中单条消息允许的最大字节数。此可调参数隐含在可调参数 *msgmnb*(5)（队列的大小）中。

有关 System V 消息队列的详细信息，请参考 *mesg*(5) 联机帮助页中的“概述”一节。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。对于 HP-UX 11i v3 及此后的版本，此参数已过时。

作者

msgmax 由 AT&T 开发。

另请参阅

msgsnd(2)、 mesg(5)、 msgmbs(5)、 msgmap(5)、 msgmnb(5)、 msgmni(5)、 msgseg(5)、 msgssz(5)、 msgtql(5)。

名称

msgmbs - 所有 System V IPC 消息队列的最大兆字节数

值

缺省值

8

保证安全

1

允许值

最小值： **1**

最大值： **0x100000**

说明

msgmbs 可调参数指定系统中任何一次的所有 System V IPC 消息允许的合计大小总数的最大值（以兆字节为单位）。

如果 **msgsnd()** 系统调用尝试超出 **msgmbs** 所强加的限制，则系统将阻塞（或者如果指定 **IPC_NOWAIT** ，则返回 [EAGAIN] ），直到 **msgrcv()** 的调用方将队列消息的字节数充分减少到新消息和其他仍在队列中的消息符合 **msgmnb** 的限制范围为止。

有关 System V 消息队列的详细信息，请参考 *mesg*(5) 联机帮助页中的“概述”一节。

更改此可调参数的人员

任何用户。

更改限制

此可调参数为动态的。对此可调参数的更改立即生效。更改会影响后续的 **msgsnd()** 操作。

应该何时增加此可调参数的值

如果应用程序需要在系统中保存更多的数据，请增加此可调参数的值。

增加此值的负面影响

增加该值将增加可能在任意时间点排队的消息数据总数。这可能导致 **msgsnd(2)** 不得不减小阻塞频率。

由于消息队列头存储在内核内存中，所以该内存不可用于其他系统服务。

应该何时降低此可调参数的值

当应用程序不需要在单个队列中保存大量的数据时，降低此可调参数的值。

降低此值的负面影响

降低该可调参数值对于活动消息没有任何影响，即使新的可调参数值小于当前排队的消息字节总数。但是，直到字节总数降低到低于 **msgmbs** 的设置后，新消息才可能开始排队。

应该同时更改的其他可调参数值

所有的 System V 消息队列的可调参数都是相互关联的，而且 不应该作为独立变量处理。此调整必须作为一个系统来评估，以保证这些可调参数能反映应用程序的要求。这些消息可调参数包括 **msgmnb** 、 **msgmni** 和 **msgtql** 可调参数。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

系统资源限制（例如，内存）可能会限制排队的消息数量和（或）总大小。这些系统限制可能会出现在 msgtql 和 msgmbs 可调参数的限制值之前。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

msgmnb 由 AT&T 开发。

另请参阅

msgrcv(2)、 msgsnd(2)、 mesg(5)、 msgmni(5)、 msgtql(5)、 msgmnb(5)。

名称

msgmnb - 单个 System V IPC 消息队列的最大字节数

值

缺省值

16384

保证安全

16384

允许值

最小值： **0**

最大值： **0x4000000**

说明

可调参数 **msgmnb** 指定在任何给定时间内，单个给定的 System V IPC 消息队列中，所有列队的消息所允许的合计大小的总数的最大值（以字节为单位）。

如果 **msgsnd()** 系统调用尝试超出 **msgmnb** 所强加的限制，则系统将阻塞（如果指定 **IPC_NOWAIT** ，则返回 [EAGAIN] ），直到 **msgrcv()** 的调用方将队列中的字节数减少到新消息和其他仍在队列中的消息低于 **msgmnb** 的限制。

有关 System V 消息队列的详细信息，请参考 *mesg*(5) 联机帮助页中的“概述”一节。

更改此可调参数的人员

任何用户。

更改限制

此可调参数是动态的。更改会影响后续的 **msgsnd()** 操作。

应该何时增加此可调参数的值

如果应用程序需要在单个的队列中保存更多的数据，请增加此可调参数的值。

应该何时降低此可调参数的值

当应用程序不需要在单个队列中保存大量的数据时，降低此可调参数的值。

应该同时更改的其他可调参数值

所有的 System V 消息队列的可调参数都是相互关联的，因此 不应该将其处理为独立的变量。集合必须作为一个系统来评估，以保证这些可调参数能反映应用程序的要求。这些消息可调参数包括 **msgmnb** 、 **msgmni** 、 **msgmbs** 和 **msgtql** 。特别是，更改可调参数 **msgmnb** 时，可调参数 **msgmbs** 和 **msgtql** 可能也需要调整。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是

缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

msgmnb 由 AT&T 开发。

另请参阅

msgrcv(2)、 msgsnd(2)、 mesg(5)、 msgmni(5)、 msgtql(5)、 msgmbs(5)。

名称

msgmni - 系统级 System V IPC 消息队列 (ID) 所允许的最大数量

值

缺省值

512

保证安全

512

允许值

最小值： **1**

最大值： **1,000,000**

说明

可调参数 **msgmni** 指定了系统级 System V IPC 消息队列标识符（每个队列对应一个标识符）的最大数量。每个新建立的队列都有一个标识符 (ID)，标识符的数量上限为 **msgmni** 。

应用程序使用 **msgget()** 系统调用创建新的队列。如果已使用所有的 ID， **msgget()** 将返回 [ENOSPC] 。

如果一个进程获取了一个消息队列，但是在进程中止时未删除它，则将保留队列以及其中的所有消息。可以使用 **ipcrm** 命令删除忽略的消息队列。可以使用 **ipcs** 命令显示消息队列的状态。

有关 System V 消息队列的详细信息，请参考 *mesg*(5) 联机帮助页中的“概述”一节。

更改此可调参数的人员

任何用户。

更改限制

此可调参数是动态的。对此可调参数的更改立即生效。

应该何时增加此可调参数的值

当应用程序需要更多的队列时，增加此可调参数的值。

应该何时降低此可调参数的值

当对队列的需求降低时，降低此可调参数的值。

降低此值的负面影响

降低此值会增加应用程序因不能创建更多消息队列而失败的风险。即使新的可调参数值小于系统中创建的队列的数量，降低此可调参数的值也不会影响任何活动消息队列。但是，在活动队列的数量降低至 **msgmni** 的设置值之前，可能无法创建新的队列。

应该同时更改的其他可调参数值

所有的 System V 消息队列的可调参数都是相互关联的，因此 不应该将其处理为独立的变量。集合必须作为一个系统来评估，以保证这些可调参数能反映应用程序的要求。这些消息可调参数包括 **msgmbs**、 **msgmnb**、 **msgmni** 和 **msgtql**。特别是，在更改可调参数 **msgmni** 时，可调参数 **msgtql** 和 **msgmbs** 可能也需要随之调整。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

msgmni 由 AT&T 开发。

另请参阅

ipcrm(1)、 ipcs(1)、 msgget(2)、 mesg(5)、 msgmnb(5)、 msgtql(5)、 msgmbs(5)。

名称

msgseg - 系统中 System V IPC 消息段的数量（已过时）

说明

msgseg 可调参数已过时并被删除。

此可调参数指定了系统级共享内存消息存储空间的“段”的总数。在所有 IPC 消息队列中都共享该空间。此可调参数已替换为可调参数 *msgmbs*(5) （待接收的消息所使用的最大内核内存，以 MB 为单位）。

有关 System V 消息队列的详细信息，请参考 *mesg*(5) 联机帮助页中的“概述”一节。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。对于 HP-UX 11i v3 及此后的版本，此参数已过时。

作者

msgseg 由 AT&T 开发。

另请参阅

msgsnd(2)、 mesg(5)、 msgmbs(5)、 msgmap(5)、 msgmax(5)、 msgmnb(5)、 msgmni(5)、 msgssz(5)、 msgtql(5)。

名称

msgssz - System V IPC 消息段字节数（已过时）

说明

msgssz 可调参数已过时并被删除。

此可调参数指定了为存储 IPC 消息而保留的内存空间“段”的大小，以字节为单位。此可调参数已替换为可调参数 *msgmbs*(5) （待接收的消息所使用的最大内核内存，以 MB 为单位）。

有关 System V 消息队列的详细信息，请参考 *mesg*(5) 联机帮助页中的“概述”一节。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。对于 HP-UX 11i v3 及此后的版本，此参数已过时。

作者

msgssz 由 AT&T 开发。

另请参阅

msgsnd(2)、 mesg(5)、 msgmbs(5)、 msgmap(5)、 msgmax(5)、 msgmnb(5)、 msgmni(5)、 msgseg(5)、 msgtql(5)。

名称

msgtql - 系统中任意时间的最大 System V IPC 消息数

值

缺省值

1024

保证安全

1024

允许值

最小值： **1**

最大值： **2147483647**

建议值

预期的最大消息数。

说明

可调参数 **msgtql** 指定了所有消息队列中系统级个体总消息数的最大值。每一条消息有一个消息头，用于指定消息类型和位置，消息头总数受 **msgtql** 的限制。

请注意，如果 **msgsnd()** 系统调用试图不受由 **msgtql** 强加的限制，则系统在消息空间可用之前，或者设定了 **IPC_NOWAIT** 的情况下，返回 [EAGAIN] 之前将一直阻塞。

其他的内核可调参数可能会限制所支持的最大消息数。这些限制因素可能是队列中的最大字节数 **msgmnb**，和系统中允许的最大总字节数 **msgmbs**。

有关 System V 消息队列的详细信息，请参考 *mesg*(5) 联机帮助页中的“概述”一节。

更改此可调参数的人员

任何用户。

更改限制

此可调参数是动态的。对此可调参数的更改立即生效。

应该何时增加此可调参数的值

如果应用程序在 **msgsnd()** 处阻塞过于频繁，内核消息队列中需要保留更多的消息总量时，增加此可调参数的值。

增加此值的负面影响

增加此可调参数的值会增加那些在任何时间点排队等待的消息的数量。这将导致 *msgsnd*(2) 不得不降低阻塞的频率。由于内核内存中存储了消息队列标头，因此其他系统服务也不能使用该内存。

应该何时降低此可调参数的值

当应用程序不再需要那么多的消息总量时，降低此可调参数的值。

降低此值的负面影响

降低此值会增加应用程序因不能发送更多消息而失败的风险。即使新的可调参数值小于排队的消息数量，降低此值也不会影响任何排队的消息。但是，在消息的数量降低至 **msgtql** 的设置值之前，任何新的消息均无法排队。

应该同时更改的其他可调参数值

所有的 System V 消息队列的可调参数都是相互关联的，因此 不应该将其处理为单独的变量。集合必须作为一个系统来评估，以保证这些可调参数能反映应用程序的要求。这些消息可调参数包括 **msgmbs** 、 **msgmnb** 、 **msgmni** 和 **msgtql** 。特别是，在调整可调参数 **msgtql** 时，可调参数 **msgmbs** 和 **msgmnb** 可能也需要随之调整。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

系统资源限制（例如，内存）可能会限制排队的消息数量和（或）总大小。这些系统限制可能会出现在 msgtql 和 msgmbs 可调参数的限制值之前。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

msgtql 由 AT&T 开发。

另请参阅

msgsnd(2)、 mesg(5)、 msgmnb(5)、 msgmni(5)、 msgmbs(5)。

名称

ncdnode - 打开 CDFS 文件的最大数量（系统级）

值

无故障

150

缺省

150

允许值

允许的最小值为 **25** 。可允许的最大值为内存容量。

指定一个正整数值。

说明

ncdnode 定义了 CD-ROM 文件系统中 i 节点表的插槽数。该数量限制了任意给定时间内 CDFS 文件系统内存中可打开的节点数。它在功能上与 **ninode** 类似，但仅应用于 CD-ROM 文件系统。

应该由谁来更改此可调参数?

当系统中运行需要访问 CD-ROM 文件系统的应用程序时，可能需要修改此可调参数。

对于更改的限制

cdfs 内核模块，用于提供对 CD-ROM 文件系统的文件系统类型的特定支持，现已是一个可动态加载的内核模块 (DLKM)。任何对可调参数 **ncdnode** 的修改都将在 **cdfs** 内核模块被卸载并重新加载后（请参阅 *kcmodule*(1M)），或在系统重新引导后生效。

何时应增加此可调参数的值?

当运行需要同时打开大量的 CDFS 文件的应用程序时可能需要增大该值。

增加此可调参数的副作用是什么?

CDFS 节点将消耗更多的系统内存。在 **cdfs** 内核模块加载或系统重新引导时，将在 **ncdnode** 值的基础上创建一个更大的 CDFS 的静态节点表。

何时应降低此可调参数的值?

要限制打开的 CDFS 文件数并减少内存消耗，应降低此值。

降低此值的副作用是什么?

在 **cdfs** 内核模块加载或系统重新引导时，CDFS 节点将消耗更少的系统内存。

同时还应更改哪些其他的可调参数值?

无。

警告

所有 HP-UX 内核可调参数都是针对于特定版本的。未来的 HP-UX 版本可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致可调参数值的更改。安装后，某些可调参数可能不再是缺

省值或建议的值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

ncdnode 由 HP 开发。

另请参阅

kctune(1M)、 kcmodule(1M)、 sam(1M)、 gettune(2)、 settune(2)、 ninode(5)。

名称

nclist - 用于 pty 和 tty 数据传输的 cblock 数量

值

缺省值

8292

允许值

最小值 **132** 。最大值受可用内存的限制。

说明

nclist 指定系统分配了多少 **cblock** 。当数据流通过 tty 和 pty 设备时， **cblock** 会存储数据流量。

nclist 的缺省值为 **8292** ，它来自这样一条规则：处理系统到控制台等的流量需要 100 个 **cblock** ，加上平均每个用户会话需要 16 个 **cblock** ，假定有 512 个用户会话。 **cblock** 同样可用于除登录会话以外的串行连接，如 SLIP 连接、UUCP 传输和终端仿真器等。如果系统正在使用这些其他种类的连接，那么请相应地增大 **nclist** 。

如果 **cblock** 池已耗尽，那么传递于 tty 或 pty 设备间的数据将丢失，因为在需要 **cblock** 时却得不到它。如果发生这种情况，则会将警告消息“cblock exhaustion has occurred n times”（请参阅 *termio(7)* ）置于系统消息缓冲区。

更改此可调参数的人员

拥有 **SYSATTR** 权限的用户。有关对支持精细划分权限的系统的授权访问的详细信息，请参阅 *privileges*(5) 。

更改限制

对此可调参数的更改将在下次重新引导时生效。

应该何时增加此可调参数的值

在以下情况中可增大可调参数值：

- 当内核发送错误消息 **WARNING: cblock exhaustion has occurred n times (see termio(7)).** 时，系统的 **cblock** 已经耗尽。这表示需要增大 **nclist** 。
- 该系统可能在运行终端 I/O 时有些慢，从而导致一些数据丢失，但没有显示警告信息。增大 **nclist** 值可解决这个问题。

nclist 的最小值为 **132** 。没有最大值，但每个 **cblock** 将消耗 32 字节的常驻（非可交换的）机器内存，因此在选择该值时应考虑到这一点。

增加此可调参数值的负面影响

将使用更多的常驻（非可交换的）机器内存。

应该何时降低此可调参数的值

在创建一个最小化系统时应降低该值。

降低此可调参数值的负面影响

系统可能耗尽 **cblocks** 。

应该同时更改的其他可调参数

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

nclist 由 HP 开发。

另请参阅

kctune(1M)、 privileges(5)、 termio(7)。

名称

ncsize - 目录名查找缓存 (DNLC) 条目数

值

无故障

5596

缺省

8976

允许值

允许的最小值为 **128** 。允许的最大值为受限的内存。对该值的另一个限制是它必须大于等于 DNLC 锁数量的八倍 **(ncsize >= 8 * dnlc_hash_lock)** 。

指定一个正整数值。

说明

为了加快在内核目录中的搜索速度，内存中有一个目录常驻缓存，称为 **Directory Name Lookup Cache (DNLC)**。在内核文件名查找过程中，任何遇到的目录或文件都会保留在 DNLC 中，以备将来引用。可调参数 **ncsize** 显示了 DNLC 的条目数，该数量在系统引导时就静态分配了。所以， **ncsize** 的值越大，内存中保存的用于文件名查找的缓存目录/文件条目数也越大。

谁需要更改这个可调参数？

HP-UX 系统管理员。

对于更改的限制

此可调参数为静态的。对该可调参数值的任何更改都需在系统重新引导后方可生效。

何时应增加此可调参数的值？

在运行需要大量访问文件名的应用程序时，可能需要增大该值以提高文件名查找性能。

增加此可调参数的副作用是什么？

DNLC 将消耗更多的系统内存。

何时应降低此可调参数的值？

可减小该值以限制 DNLC 中的条目数并减少内存消耗。

降低此值的副作用是什么？

DNLC 在系统重新引导时将消耗更少的系统内存。需要文件名查找的操作（例如 open(2)）可能遇到性能降低的情况。

同时还应更改哪些其他可调参数的值？

ncsize 的值必须大于等于 **dnlc_hash_locks** 值的八倍 **(ncsize >= 8 * dnlc_hash_lock)** 。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。未来的 HP-UX 版本可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

ncsize 由 HP 开发。

另请参阅

kctune(1M)、 sam(1M)、 gettune(2)、 settune(2)、 dnlc_hash_locks(5)。

名称

nfile - 打开文件的最大数量（系统级）

值

保证安全

0

缺省值

nfile 的缺省值为 **0** （零）。值为零意味着通常由 **nfile** 实施的系统限制将被禁用（即，系统级打开文件的数量仅受可用内存的限制）。

允许值

nfile 的允许值为 **0** （零），或为最小值和最大值之间（包含边界值）的值。最小值为 **2048** ，最大值为系统可表示的最大的 32 位带符号整数值。对该值的另一个限制是它必须大于或等于每个进程打开文件的硬限制的两倍（即 **nfile >= 2*maxfiles_lim** ）。

指定一个正整数值。

说明

现在，此可调参数是专用的并且不建议使用。不应该再使用它。在 11iV3 中，已重新构建系统打开文件表，从而删除了对该值的体系结构限制，因此， **nfile** 不再是必需的。要控制在系统上打开文件的最大数量，建议的方法是适当地设置 **maxfiles_lim** 和 **nproc** 的值；理论上，系统最大数可以假定为 **maxfiles_lim** * **nproc**。

可调参数 **nfile** 定义了系统打开文件表中的最大插槽数。该数值限制了系统中所有进程累积的打开文件数。除了指定的文件（常规文件、目录、链接、设备文件等等）外，其他占用系统打开文件表中的插槽的对象包括管道、FIFO、套接字和流。请注意， **dup** 和 **dup2** 系统调用尽管消耗每个进程文件表的条目，但不占用系统打开文件表的新插槽。

更改此可调参数的人员

此可调参数将不进行更改。

更改限制

该可调参数是动态的；对它的调整将在运行的系统上立即生效。动态调整 **nfile** 的值时，不可将其设置为低于当前运行内核中打开的文件数的值。要使系统级打开文件的数量不受限制，应将此值设置为缺省值。

应该何时增加此可调参数的值

要实施对打开文件数的限制，应将此可调参数设置为非零值。

增加此值的负面影响

通过将此可调参数设置为正的非零值，将强制执行系统级限制。通过要求系统实施全局限制， **open()** 系统调用和其他相关系统调用的性能可能会受到影响。

应该何时降低此可调参数的值

很少需要减小 **nfile** 的值。可减小该值以限制系统中打开的文件数量，这样可减少内存消耗。

降低此值的负面影响

较低的限制可以限制应用程序分配新文件描述符或打开指定文件的能力。

应该同时更改的其他可调参数值

nfile 必须大于等于 **maxfiles_lim** 值的两倍。内核在可调参数设置期间检查以确保这一点。

警告

此可调参数已过时，会从将来的 HP-UX 发行版中删除。

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

nfile 由 HP 开发。

另请参阅

kctune(1M)、 sam(1M)、 dup(2)、 dup2(2)、 gettune(2)、 open(2)、 settune(2)、 maxfiles(5)、 maxfiles_lim(5)、 nproc(5)。

名称

nflocks - 文件锁的最大数量

值

无故障

1024

缺省

缺省值是在运行时计算出来的，它取决于系统物理内存量。对于小内存系统（小于 1GB），缺省值为 1200。对于内存大于 1GB 的系统，缺省值为 4096，或 4K。

允许值

最小值为 50。最大值为 0x1000000。

指定一个正整数值。

说明

可调参数 **nflocks** 表示系统级可用的文件锁的最大数量。

谁需要更改这个可调参数？

系统要运行需大量文件锁的应用程序时应更改此可调参数。更改该值时请注意，一个文件可能有多个锁，而使用了 **lockf()** 或 **fcntl()** 的数据库可能需要极其大量的锁。

对于更改的限制

该可调参数是动态的（调整将在运行的系统上立即生效）。尽管 **nflocks** 的值可以动态调整，但不可将其设为低于当前运行内核中文件锁数量的值。

何时应增加此可调参数的值？

当需要同时使用大量文件锁时应增大该值。

增加此可调参数的副作用是什么？

内核将按比例为新的 **nflocks** 值分配一部分内存，并按比例释放旧值的内存。创建新锁将消耗更多的内存。

何时应降低此可调参数的值？

可减小该值以限制系统中可用文件锁的数量并减少内存的消耗。

降低此值的副作用是什么？

内核将按比例为新的 **nflocks** 值分配一部分内存，并按比例释放旧值的内存。如果旧值的使用率高于该新值，那么多出的内存可释放至系统以用于其他目的。

同时还应更改哪些其他可调参数的值？

无。

警告

将 **nflocks** 设为最大值 (0x1000000) 将导致大量内存的分配 (˜.5GB)。如果没有足够可用内存，或者剩余的可用内存过少以至可能影响系统性能，在 [ENOMEM] 设置下将无法满足更改 **nflocks** 值的要求。

所有 HP-UX 内核可调参数都是针对于特定发行版的。未来的 HP-UX 版本可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

nflocks 由 HP 开发。

另请参阅

kctune(1M)、 sam(1M)、 gettune(2)、 settune(2)、 fcntl(2)、 lockf(2)。

名称

nfs2_max_threads - 控制执行 NFS 第 2 版客户端异步 I/O 的内核线程的数量

值

保证安全

16

缺省值

16

允许值

最小值： **0**

最大值： **nkthread/5**

建议值

如果此可调参数的设置值大于 256 个线程，则在运行时发出警告，原因是该设置值已超出测试的限制。这不是一个严重的警告，而只是提供给管理员的一条消息。

说明

nfs2_max_threads 控制执行 NFS 第 2 版客户端异步 I/O 的内核线程的数量。由于 NFS 是基于 RPC 的，而 RPC 本质上是同步的，因此执行与调用线程异步的 NFS 操作时需要单独的执行相关环境。对于 read-ahead，可以异步执行的操作是读取；对于 readdir read-ahead，可以异步执行的操作是 readdir；而对于 putpage 和 pageio 请求则是写入。

更改此可调参数的人员

分布式文件系统管理员应该根据可用的网络带宽检查此值。

更改限制

nfs2_max_threads 可调参数是动态的；但是，文件系统的线程数量是在挂接文件系统时设置的。要影响某个特定的文件系统，请在更改此参数之后卸除并重新挂接该文件系统。

应该何时增加此可调参数的值

如果网络带宽非常高，并且客户端和服务器具有充足的资源，则增加此值可以更加有效地利用可用的网络带宽、客户端资源以及服务器资源。

增加此值的负面影响

增加系统资源，可能需要增加 **nkthread** （请参阅 *nkthread*(5)）。用于 NFS 第 2 版和 NFS 第 3 版挂接点的异步线程总数不能超过 **nkthread** 定义的可用线程总数的 20%。如果 NFS 挂接命令无法保证能够为该挂接点创建最大数量的线程，则 NFS 挂接将失败。

应该何时降低此可调参数的值

在网络带宽非常低的情况下，请降低此值，以便 NFS 客户端不会使网络超载。

降低此值的负面影响

降低此值可能会影响 NFS 性能。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

nfs2_max_threads 由 Sun Microsystems, Inc. 开发。

另请参阅

kctune(1M)、 sam(1M)、 gettune(2)、 settune(2)、 nfs3_max_threads(5)、 nfs4_max_threads(5)、 nkthread(5)。

名称

nfs2_nra - 按顺序访问文件时，用于控制 NFS 第 2 版客户端排队的预读操作数

值

保证安全

4

缺省值

4

允许值

最小值： **0**

最大值： **MAXINT**

建议值

如果此可调参数的设置值大于 16，则在运行时发出警告，原因是该设置值已超出测试的限制。

这不是严重警告，而只是提供给管理员的一条消息。

说明

nfs2_nra 可控制发现对文件的顺序访问后，由 NFS 第 2 版客户端排队的预读操作数。这些预读操作可增加并发性和读取吞吐量。通常，每个预读请求都用于获取 8192 字节的文件数据。

更改此可调参数的人员

分布式文件系统管理员应该根据网络带宽和客户端的内存压力检查此值。

更改限制

nfs2_nra 可调参数是动态的；所做的调整将在运行的系统上立即生效。

应该何时增加此可调参数的值

如果网络带宽非常高，并且客户端和服务器具有充足的资源，则增加此值可以更加有效地利用可用的网络带宽、客户端资源以及服务器资源。

增加此值的负面影响

基于网络带宽进行不当的调整可能会导致性能问题。

应该何时降低此可调参数的值

在网络带宽非常低的情况下，可能需要降低此值，以便 NFS 客户端不会使网络超载。

降低此值的负面影响

基于网络带宽进行不当的调整可能会导致性能问题。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是

缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

nfs2_nra 由 Sun Microsystems, Inc. 开发。

另请参阅

kctune(1M)、 sam(1M)、 gettune(2)、 settune(2)、 nfs3_nra(5)、 nfs4_nra(5)、 values(5)。

名称

nfs3_bsize - 控制 NFS 第 3 版客户端使用的逻辑块大小

值

保证安全

32768

缺省值

32768

允许值

最小值： **4096**

最大值： **MAXINT**

建议值

如果此可调参数的设置值大于 1048576 字节，则运行时将发出警告，原因是该设置值已超出测试的限制。这不是严重警告，而只是提供给管理员的一条消息。

说明

nfs3_bsize 可控制 NFS 第 3 版客户端使用的逻辑块大小。此块大小代表客户端执行 I/O 时，尝试在服务器中读取或写入的数据量。

更改此可调参数的人员

分布式系统管理员应检查此值，并在需要更大传输大小时更改此值。

更改限制

nfs3_bsize 可调参数是动态的；但是，挂接了文件系统后，将会设置文件系统的逻辑块大小设置。要影响特定的文件系统，必须在更改此参数后取消挂接该文件系统，然后重新挂接。

应该何时增加此可调参数的值

如果需要更大的传输，请增加此参数以及 **nfs3_max_transfer_size** 和 **nfs3_max_transfer_size_cots** 参数。

增加此值的负面影响

将使用更多的内核内存来管理传输缓冲区。

应该何时降低此可调参数的值

如果需要更小的传输大小，请降低此可调参数的值。

降低此值的负面影响

将此值设置为小于 32 KB 会增加向 NFS 服务器发出的请求数。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是

缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

nfs3_bsize 由 Sun Microsystems, Inc. 开发。

另请参阅

kctune(1M)、 sam(1M)、 gettune(2)、 settune(2)、 nfs3_max_transfer_size(5)、 nfs3_max_transfer_size_cots(5)、 nfs4_bsize(5)、 values(5)。

名称

nfs3_do_readdirplus - 在 NFS 服务器上打开或关闭 NFS 第 3 版 readdirplus 功能

值

保证安全

1

缺省值

1

允许值

0：关闭 NFS 第 3 版 readdirplus 功能

1：打开 NFS 第 3 版 readdirplus 功能

说明

nfs3_do_readdirplus 可控制在 NFS 服务器上打开或关闭 NFS 第 3 版 readdirplus 功能的能力。如果关闭此功能，将强制客户端改用 readdir 功能。

更改此可调参数的人员

如果在读取较大目录时遇到性能问题，分布式文件系统管理员应检查此值。

更改限制

nfs3_do_readdirplus 可调参数是动态的；所做的调整将在运行的系统上立即生效。

关闭 **Readdirplus** 功能的负面影响

将强制客户端使用 readdir 代替 readdirplus，客户端将不再遵循 NFS 第 3 版协议。如果出现问题，请重新打开 **nfs3_do_readdirplus** 。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

nfs3_do_readdirplus 由 HP 开发。

另请参阅

kctune(1M)、sam(1M)、 gettune(2)、settune(2)。

名称

nfs3_jukebox_delay - 控制 NFS 第 3 版客户端在收到 NFS3ERR_JUKEBOX 错误后重新传输请求之前等待的时间

值

保证安全

1000

缺省值

1000

允许值

最小值： **100** ，大约为一秒钟

最大值： **MAXINT**

建议值

如果此可调参数的设置值小于 1 秒的 100% 或大于 1 秒的 60000%，则在运行时发出警告，原因是该设置值已超出测试的限制。这不是一个严重的警告，而只是提供给管理员的一条消息。

说明

nfs3_jukebox_delay 控制 NFS 第 3 版客户端在收到来自前面请求的错误 **NFS3ERR_JUKEBOX** 之后，传输新请求之前所等待的时间。错误 **NFS3ERR_JUKEBOX** 通常是在由于某种原因文件暂时不可用时从服务器返回的。这些情况通常与分层存储设备和 CD 或磁带自动唱片点唱机相关。

更改此可调参数的人员

分布式文件系统管理员应根据 NFS 文件服务器上的存储设备的速度来检查该值。

更改限制

nfs3_jukebox_delay 可调参数是动态的，所做的调整将在运行系统上立即生效。

应该何时增加此可调参数的值

如果在使文件可用时出现长时间延迟，则应增加该值。由于重复进行重新传输，因此将减少网络开销。

增加此值的负面影响

如果文件不可用的时间比增加的延迟时间短，则增加重新传输之前的延迟会影响性能。

应该何时降低此可调参数的值

如果文件不可用的时间很短，则降低该值不会影响性能。

降低此值的负面影响

如果文件不可用的时间很长，则会增加网络开销。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是

缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

nfs3_jukebox_delay 由 Sun Microsystems, Inc. 开发。

另请参阅

kctune(1M)、 sam(1M)、 gettune(2)、 settune(2)、 values(5)。

名称

nfs3_max_threads - 控制为 NFS 第 3 版客户端执行异步 I/O 的内核线程数

值

保证安全

16

缺省值

16

允许值

最小值： **0**

最大值： **nkthread/5**

建议值

如果此可调参数的设置值大于 256 个线程，则在运行时发出警告，原因是该设置值已超出测试的限制。这不是一个严重的警告，而只是提供给管理员的一条消息。

说明

nfs3_max_threads 控制执行 NFS 第 3 版客户端异步 I/O 的内核线程的数量。由于 NFS 是基于 RPC 的，而 RPC 本质上是同步的，因此执行与调用线程异步的 NFS 操作时需要单独的执行相关环境。对于 read-ahead，可以异步执行的操作是读取；对于 readdir read-ahead，可以异步执行的操作是 readdir；而对于 putpage 和 pageio 请求则是写入。

更改此可调参数的人员

分布式文件系统管理员应该根据可用的网络带宽检查此值。

更改限制

nfs3_max_threads 可调参数是动态的；但是，文件系统的线程数量是在挂接文件系统时设置的。要影响某个特定的文件系统，请在更改此参数之后卸除并重新挂接该文件系统。

应该何时增加此可调参数的值

如果网络带宽非常高，并且客户端和服务器具有充足的资源，则增加此值可以更加有效地利用可用的网络带宽、客户端资源以及服务器资源。

增加此值的负面影响

增加系统资源，可能需要增加 **nkthread** （请参阅 *nkthread*(5) ）。用于 NFS 第 2 版和 NFS 第 3 版挂接点的异步线程总数不能超过 **nkthread** 定义的可用线程总数的 20%。如果 NFS 挂接命令无法保证能够为该挂接点创建最大数量的线程，则 NFS 挂接将失败。

应该何时降低此可调参数的值

在网络带宽非常低的情况下，请降低此值，以便 NFS 客户端不会使网络超载。

降低此值的负面影响

降低此值可能会影响 NFS 性能。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

nfs3_max_threads 由 Sun Microsystems, Inc. 开发。

另请参阅

kctune(1M)、 sam(1M)、 gettune(2)、 settune(2)、 nfs2_max_threads(5)、 nfs4_max_threads(5)、 nkthread(5)。

名称

nfs3_max_transfer_size - 控制 NFS 第 3 版 read、write、readdir 或 readdirplus 请求的数据部分大小

值

保证安全

1048576

缺省值

1048576

允许值

最小值：**4096**

最大值：**MAXINT**

建议值

如果此可调参数的设置值大于 1048576，则在运行时发出警告，原因是该设置值已超出测试的限制。这不是一个严重的警告，而只是提供给管理员的一条消息。

说明

nfs3_max_transfer_size 控制 NFS 第 3 版读取、写入、 readdir 或 readdirplus 请求的最大数据部分大小。此参数控制服务器返回的请求的最大大小，以及客户端生成的请求的最大大小。

更改此可调参数的人员

如果分布式文件系统管理员需要减少或增加 NFS 数据包的缺省大小，则应该检查此值。

更改限制

nfs3_max_transfer_size 可调参数是动态的；但是，挂接了文件系统后，将会设置文件系统的传输大小。要影响特定的文件系统，请在更改此参数后，取消挂接该文件系统，然后重新挂接。

NFS 第 3 版实际传输大小由下列可调参数控制： **nfs3_bsize** 、 **nfs3_max_transfer_size** 和 **nfs3_max_transfer_size_cots** 。实际传输大小取决于具有最小值的可调参数。对于 NFS 第 3 版 TCP 流量，请将 **nfs3_max_transfer_size** 、 **nfs3_max_transfer_size_cots** 和 **nfs3_bsize** 增加到相同值，以增加传输大小。要减小传输大小，更改 **nfs3_bsize** 即可。

应该何时增加此可调参数的值

增加传输大小会减少向 NFS 服务器发送的请求数，但是 NFS 客户端和服务器上将会消耗更多的系统资源。

增加此值的负面影响

将限制设置得过大可能会导致 UDP 错误，并可能导致客户端和服务器之间的通信出现问题。

应该何时降低此可调参数的值

如果 NFS 通信失败，则应将此值改回为缺省值。

降低此值的负面影响

由于 I/O 请求数增加，NFS 第 3 版网络流量将会增加。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

nfs3_max_transfer_size 由 Sun Microsystems, Inc. 开发。

另请参阅

kctune(1M)、 sam(1M)、 gettune(2)、 settune(2)、 nfs3_bsize(5)、 nfs3_max_transfer_size_cots(5)、 values(5)。

名称

nfs3_max_transfer_size_cots - 控制 NFS 第 3 版通过 TCP 发出的 read、write、readdir 或 readdirplus 请求的数据部分大小

值

保证安全

1048576

缺省值

1048576

允许值

最小值： **4096**

最大值： **MAXINT**

建议值

如果此可调参数的设置值大于 1048576，则在运行时发出警告，原因是该设置值已超出测试的限制。这不是一个严重的警告，而只是提供给管理员的一条消息。

说明

nfs3_max_transfer_size_cots 控制 NFS 第 3 版通过 TCP 发送的 read、write、 readdir 或 readdirplus 请求的数据部分的最大大小。此参数既控制服务器返回的请求的最大大小，还控制客户端生成的请求的最大大小。

更改此可调参数的人员

如果分布式系统管理员要减少或增加 NFS 第 3 版通过 TCP 发送的数据包的缺省大小，则应检查此值。

更改限制

nfs3_max_transfer_size_cots 可调参数是动态的，但是，文件系统的传输大小是在挂接文件系统时设置的。要影响特定的文件系统，请在更改此参数后卸除并挂接文件系统。

NFS 第 3 版通过 TCP 实际传输的大小由以下可调参数控制： **nfs3_bsize** 、 **nfs3_max_transfer_size** 和 **nfs3_max_transfer_size_cots** 。实际传输大小取决于具有最小值的可调参数。对于 NFS TCP 流量，将 **nfs3_max_transfer_size** 、 **nfs3_max_transfer_size_cots** 和 **nfs3_bsize** 增加到相同的值会增加传输大小。要减小传输大小，则只须更改 **nfs3_bsize** 即可。

应该何时增加此可调参数的值

增加传输大小会减少向 NFS 服务器发送的请求数，但是 NFS 客户端和服务器上将会消耗更多的系统资源。

增加此值的负面影响

设置的限制过大将导致 NFS 客户端和 NFS 服务器消耗更多的系统资源。

应该何时降低此可调参数的值

如果 NFS 通信失败，则应将此值改回为缺省值。

降低此值的负面影响

增加 I/O 请求的数目将导致增加 NFS 第 3 版通过 TCP 的网络流量。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

nfs3_max_transfer_size 由 Sun Microsystems, Inc. 开发。

另请参阅

kctune(1M)、 sam(1M)、 gettune(2)、 settune(2)、 nfs3_bsize(5)、 nfs3_max_transfer_size(5)、 values(5)。

名称

nfs3_nra - 按顺序访问文件时，控制由 NFS 第 3 版客户端排队的预读操作数

值

保证安全

4

缺省值

4

允许值

最小值： **0**

最大值： **MAXINT**

建议值

如果此可调参数的设置值大于 16，则在运行时发出警告，原因是该设置值已超出测试的限制。这不是一个严重的警告，而只是提供给管理员的一条消息。

说明

nfs3_nra 控制由 NFS 第 3 版客户端排队的预读操作的数目（当发现按顺序访问文件时）。这些预读操作增加并发和读取数据流量。每个预读请求通常适用于 32768 字节的文件数据。

更改此可调参数的人员

分布式文件系统管理员应该根据网络带宽和客户端的内存压力检查此值。

更改限制

nfs3_nra 可调参数是动态的（所做的调整将在运行系统上立即生效）。

应该何时增加此可调参数的值

如果网络带宽非常高，并且客户端和服务器具有充足的资源，则增加此值可以更加有效地利用可用的网络带宽、客户端资源以及服务器资源。

增加此值的负面影响

基于网络带宽进行不当的调整可能会导致性能问题。

应该何时降低此可调参数的值

在网络带宽非常低的情况下，请降低此值，以便 NFS 客户端不会使网络超载。

降低此值的负面影响

基于网络带宽进行不当的调整可能会导致性能问题。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

nfs3_nra 由 Sun Microsystems, Inc. 开发。

另请参阅

kctune(1M)、 sam(1M)、 gettune(2)、 settune(2)、 nfs2_nra(5)、 nfs4_nra(5)、 values(5)。

名称

nfs4_bsize - 控制 NFS 第 4 版客户端使用的逻辑块大小

值

保证安全

32768

缺省值

32768

允许值

最小值： **4096**

最大值： **MAXINT**

建议值

如果此可调参数的设置值大于 1048576 字节，则在运行时发出警告，原因是该设置值已超出测试的限制。这不是一个严重的警告，而只是提供给管理员的一条消息。

说明

nfs4_bsize 控制 NFS 第 4 版客户端使用的逻辑块的大小。此块大小表示在客户端在需要执行 I/O 操作时，从服务器读取或写入服务器的数据量。

更改此可调参数的人员

分布式系统管理员应检查此值，并在需要更大传输大小时更改此值。

更改限制

nfs4_bsize 可调参数是动态的；不过，文件系统的逻辑块大小设置是在挂接文件系统时设置的。要影响特定的文件系统，请在更改此参数后卸除和挂接文件系统。

应该何时增加此可调参数的值

如果需要更大的传输，则增大此参数以及 **nfs4_max_transfer_size** 和 **nfs4_max_transfer_size_cots** 参数的大小。

增加此值的负面影响

将使用更多的内核内存来管理传输缓冲区。

应该何时降低此可调参数的值

如果需要更小的传输大小，请降低此可调参数的值。

降低此值的负面影响

将此值设置为小于 32 KB 将会增加对 NFS 服务器的请求数。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

nfs4_bsize 由 Sun Microsystems, Inc. 开发。

另请参阅

kctune(1M)、 sam(1M)、 gettune(2)、 settune(2)、 nfs3_bsize(5)、 nfs4_max_transfer_size(5)、 nfs4_max_transfer_size_cots(5)、 values(5)。

名称

nfs4_max_threads - 控制执行 NFS 第 4 版客户端异步 I/O 的内核线程数

值

保证安全

16

缺省值

16

允许值

最小值： **0**

最大值： **nkthread/5**

建议值

如果此可调参数的设置值大于 256 个线程，则在运行时发出警告，原因是该设置值已超出测试的限制。这不是一个严重的警告，而只是提供给管理员的一条消息。

说明

nfs4_max_threads 控制执行 NFS 第 4 版客户端异步 I/O 的内核线程数。由于 NFS 基于 RPC，而 RPC 本质上是同步的，因此，要执行与调用线程异步的 NFS 操作，需要单独的执行环境。可以异步执行的操作包括 read（代表 read-ahead）、readdir（代表 readdir read-ahead）和 write（代表 putpage 和 pageio 请求）。

更改此可调参数的人员

分布式文件系统管理员应该根据可用的网络带宽检查此值。

更改限制

nfs4_max_threads 可调参数是动态的；然而，当文件系统已挂接时，该文件系统的线程数则是固定的。要影响特定的文件系统，请在更改此参数后，卸除并挂接该文件系统。

应该何时增加此可调参数的值

如果网络带宽非常高，并且客户端和服务器具有充足的资源，则增加此值可以更加有效地利用可用的网络带宽、客户端资源以及服务器资源。

增加此值的负面影响

系统资源增加，并且可能需要增加 **nkthread** （请参阅 *nkthread*(5)）。NFS 第 4 版挂接点的异步线程总数不能超过 **nkthread** 定义的可用线程的 20%。如果 NFS 挂接命令无法保证能够为该挂接点创建最大数量的线程，则 NFS 挂接将失败。

应该何时降低此可调参数的值

在网络带宽非常低的情况下，请降低此值，以便 NFS 客户端不会使网络超载。

降低此值的负面影响

降低此值可能会影响 NFS 性能。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

nfs4_max_threads 由 Sun Microsystems, Inc. 开发。

另请参阅

kctune(1M)、 sam(1M)、 gettune(2)、 settune(2)、 nfs2_max_threads(5)、 nfs3_max_threads(5)、 nkthread(5)。

名称

nfs4_max_transfer_size - 控制 NFS 第 4 版 read、write、readdir 或 readdirplus 请求的数据部分大小

值

保证安全

1048576

缺省值

1048576

允许值

最小值： **4096**

最大值： **MAXINT**

建议值

如果此可调参数的设置值大于 1048576，则在运行时发出警告，原因是该设置值已超出测试的限制。这不是一个严重的警告，而只是提供给管理员的一条消息。

说明

nfs4_max_transfer_size 控制 NFS 第 4 版 read、write、readdir 或 readdirplus 请求的数据部分的最大大小。此参数既控制服务器返回的请求的最大大小，也控制客户端生成的请求的最大大小。

更改此可调参数的人员

如果分布式文件系统管理员需要减少或增加 NFS 数据包的缺省大小，则应该检查此值。

更改限制

nfs4_max_transfer_size 可调参数是动态的；但是，当文件系统已挂接时，该文件系统的传输大小则是固定的。要影响特定的文件系统，请在更改此参数后，卸除并挂接该文件系统。

实际 NFS 第 4 版传输大小由这些可调参数控制： **nfs4_bsize** 、 **nfs4_max_transfer_size** 和 **nfs4_max_transfer_size_cots** 。实际传输大小将取决于具有最小值的可调参数。对于 NFS 第 4 版流量，将 **nfs4_max_transfer_size** 、 **nfs4_max_transfer_size_cots** 和 **nfs4_bsize** 增加到相同的值，以增加传输大小。要降低传输大小，更改 **nfs4_bsize** 即可。

应该何时增加此可调参数的值

增加传输大小会减少向 NFS 服务器发送的请求数，但是 NFS 客户端和服务器上将会消耗更多的系统资源。

增加此值的负面影响

增加 NFS 客户端和 NFS 服务器的系统资源。

应该何时降低此可调参数的值

如果 NFS 通信失败，则应将此值改回为缺省值。

降低此值的负面影响

由于 I/O 请求数量增加， NFS 第 4 版网络流量将增加。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

nfs4_max_transfer_size 由 Sun Microsystems, Inc. 开发。

另请参阅

kctune(1M)、 sam(1M)、 gettune(2)、 settune(2)、 nfs4_bsize(5)、 nfs4_max_transfer_size_cots(5)、 values(5)。

名称

nfs4_max_transfer_size_cots - 控制 TCP 上 NFS 第 4 版 read、write、readdir 或 readdirplus 请求的数据部分大小

值

保证安全

1048576

缺省值

1048576

允许值

最小值： **4096**

最大值： **MAXINT**

建议值

如果此可调参数的设置值大于 1048576，则在运行时发出警告，原因是该设置值已超出测试的限制。这不是一个严重的警告，而只是提供给管理员的一条消息。

说明

nfs4_max_transfer_size_cots 控制 TCP 上 NFS 第 4 版 read、write、readdir 或 readdirplus 请求的数据部分的最大大小。此参数既控制服务器返回的请求的最大大小，也控制客户端生成的请求的最大大小。

更改此可调参数的人员

如果分布式文件系统管理员希望减少或增加 TCP 上的 NFS 第 4 版数据包的缺省大小，则应该检查此值。

更改限制

nfs4_max_transfer_size_cots 可调参数是动态的；但是，当文件系统已挂接时，该文件系统的传输大小则是固定的。要影响特定的文件系统，请在更改此参数后，卸除并挂接该文件系统。

TCP 上实际 NFS 第 4 版传输大小由这些可调参数控制： **nfs4_bsize** 、 **nfs4_max_transfer_size** 和 **nfs4_max_transfer_size_cots** 。实际传输大小将取决于具有最小值的可调参数。对于 NFS TCP 流量，应该将 **nfs4_max_transfer_size** 、 **nfs4_max_transfer_size_cots** 和 **nfs4_bsize** 增加到相同的值，以增加传输大小。要减小传输大小，只需更改 **nfs4_bsize** 。

应该何时增加此可调参数的值

增加传输大小会减少向 NFS 服务器发送的请求数，但是 NFS 客户端和服务器上将会消耗更多的系统资源。

增加此值的负面影响

设置的限制过大将导致 NFS 客户端和 NFS 服务器消耗更多的系统资源。

应该何时降低此可调参数的值

如果 NFS 通信失败，则应将此值改回为缺省值。

降低此值的负面影响

由于 I/O 请求数量增加， NFS 第 4 版网络流量将增加。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

nfs4_max_transfer_size 由 Sun Microsystems, Inc. 开发。

另请参阅

kctune(1M)、 sam(1M)、 gettune(2)、 settune(2)、 nfs4_bsize(5)、 nfs4_max_transfer_size(5)、 values(5)。

名称

nfs4_nra - 按顺序访问文件时，控制由 NFS 第 4 版客户端排队的预读操作数

值

保证安全

4

缺省值

4

允许值

最小值： **0**

最大值： **MAXINT**

建议值

如果此可调参数的设置值大于 16，则在运行时发出警告，原因是该设置值已超出测试的限制。这不是一个严重的警告，而只是提供给管理员的一条消息。

说明

在发现对文件的顺序访问时， **nfs4_nra** 控制由 NFS 第 4 版客户端排队的预读操作数。这些预读操作会增加并发性并读取吞吐量。通常，每个预读请求均用于获取 32768 字节的文件数据。

更改此可调参数的人员

分布式文件系统管理员应该根据网络带宽和客户端的内存压力检查此值。

更改限制

nfs4_nra 可调参数是动态的；所做的调整将在运行的系统上立即生效。

应该何时增加此可调参数的值

如果网络带宽非常高，并且客户端和服务器具有充足的资源，则增加此值可以更加有效地利用可用的网络带宽、客户端资源以及服务器资源。

增加此值的负面影响

基于网络带宽进行不当的调整可能会导致性能问题。

应该何时降低此可调参数的值

在网络带宽非常低的情况下，请降低此值，以便 NFS 客户端不会使网络超载。

降低此值的负面影响

基于网络带宽进行不当的调整可能会导致性能问题。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是

缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

nfs4_nra 由 Sun Microsystems, Inc. 开发。

另请参阅

kctune(1M)、 sam(1M)、 gettune(2)、 settune(2)、 nfs2_nra(5)、 nfs3_nra(5)、 values(5)。

名称

nfs_portmon - 启用（或禁用）NFS 服务器的源端口验证检查

值

保证安全

0

缺省值

0

允许值

最小值： **0** - 禁用检查

最大值： **1** - 启用检查

说明

nfs_portmon 可控制 NFS 服务器为了对其客户端组成部分实施完整性而执行的某些安全性检查。NFS 服务器可以查看发送请求的源端口是否为保留端口；保留端口是指端口号小于 1024 的端口。对于基于 BSD 的系统，这些端口是为有权限的用户正在运行的进程而保留的。此检查功能有助于防止用户编写自己的基于 RPC 的应用程序，从而使 NFS 服务器使用的访问检查无效。

更改此可调参数的人员

如果分布式文件系统管理员希望防止恶意用户通过使用 NFS 服务器访问在一般情况下无法访问的文件，则应检查此参数的值。

更改限制

nfs_portmon 可调参数是动态的；任何更改将在运行的系统上立即生效。

并非到处都支持保留端口的概念。因此，如果启用此检查，可能会导致互操作性问题。

启用此检查的负面影响

某些 NFS 客户端可能无法连接到 NFS 服务器。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

nfs_portmon 由 Sun Microsystems, Inc. 开发。

另请参阅

kctune(1M)、 sam(1M)、 gettune(2)、 settune(2)。

名称

nfssec - NFS 安全模式概述

说明

mount_nfs(1M) 和 *share_nfs*(1M) 命令都提供了一种方法，来指定要通过 **sec=***mode* 选项在 NFS 文件系统上使用的安全模式。 *mode* 可以是 **sys** 、 **dh** 、 **krb5** 、 **krb5i** 、 **krb5p** 或 **none** 。还可以将这些安全模式添加到自动挂接映射。请注意， *mount_nfs*(1M) 和 *automount*(1M) 此时不支持 **sec=***none* 。

share_nfs(1M) 命令行上的 **sec=***mode* 选项用于建立 NFS 服务器的安全模式。如果 NFS 连接使用 NFS 第 3 版协议， NFS 客户端必须查询服务器以寻找要使用的适用 *mode* 。如果 NFS 连接使用 NFS 第 2 版协议，则 NFS 客户端将使用缺省安全模式，当前的缺省安全模式为 **sys** 。NFS 客户端可以通过在命令行上指定 **sec=**mode 选项，来强制使用特定的安全模式。但是，如果服务器上的文件系统没有共享该安全模式，客户端可能被拒绝访问。

如果 NFS 客户端需要使用特定（更强）的安全模式对 NFS 服务器进行身份验证，则客户端需要指定要使用的安全模式，即使连接使用 NFS 第 3 版协议也是如此。这可以保证冒充服务器的攻击者不会危及客户端的安全。

下面对 NFS 安全模式进行了说明。在这些模式中， **krb5** 、 **krb5i** 和 **krb5p** 模式使用 Kerberos V5 协议对共享文件系统进行身份验证和保护。使用这些模式之前，必须将系统配置为 Kerberos 领域的一部分。

sys 使用 **AUTH_SYS** 身份验证。系统以明文形式在网络上传递用户的 UNIX 用户 ID 和组 ID， NFS 服务器不对其进行身份验证。这是最简单的安全方法，不需要其他管理操作。它是 HP-UX NFS 第 2 版客户端和 HP-UX NFS 服务器所使用的缺省方法。

dh 使用 Diffie-Hellman 公用密钥系统（ AUTH_DES ，在即将推出的 Internet RFC 中称为 **AUTH_DH**）。

krb5 在授予访问共享文件系统的权限之前使用 Kerberos V5 协议对用户进行身份验证。

krb5i 将 Kerberos V5 身份验证与完整性检查（校验和）结合使用，以验证数据是否未被篡改。

krb5p 在共享文件系统上的用户 Kerberos V5 身份验证、完整性校验和及隐私保护（加密）。由于对所有流量都进行了加密，因此这种方法可以提供最安全的文件系统共享。需要注意的是，使用 **krb5p** 时某些系统上的性能可能会受到影响，具体取决于加密算法的计算能力以及所传输的数据量。

无 使用空身份验证 (**AUTH_NONE**) 。使用 **AUTH_NONE** 的 NFS 客户端没有标识，这些客户端将由 NFS 服务器映射到匿名用户 **nobody** 。如果客户端使用的安全模式不同于 HP-UX NFS 服务器用来共享文件系统的安全模式，则该客户端会将其安全模式映射到 **AUTH_NONE** 。在这种情况下，如果与 **sec=none** 共享文件系统，则来自客户端的用户将被映射至匿名用户。

警告

/etc/nfssec.conf 会列出 NFS 安全服务。请勿编辑此文件。用户不能对其进行配置。

文件

/etc/nfs/nfsec.conf NFS 安全服务配置文件

另请参阅

automount(1M)、 mount_nfs(1M)、 share_nfs(1M)、 rpc_clnt_auth(3N)、 secure_rpc(3N)、 nfssec.conf(4)。

名称

ninode - 内存中 HFS 文件系统打开 i 节点的最大数量

值

无故障

476

缺省

缺省值是在运行时计算出来的，它取决于系统物理内存量。对于小内存系统（小于 1GB），缺省值为 4880。对于内存大于 1GB 的系统，缺省值为 8192，或 8K。

允许值

可允许的最小值为 14。可允许的最大值为内存容量。

指定一个正整数值。

说明

可调参数 **ninode** 定义了 HFS 的 i 节点表中的插槽数。该数量限制了任意指定时间 HFS 文件系统在内存中可打开的 i 节点数量。 i 节点表用作一个缓冲内存。考虑性能原因，最近的 **ninode** 打开的 i 节点（序号）保留在主存中。该表是散列表。

每个单独的打开文件都有一个打开的 i 节点与其相关联。因此，单独的打开文件数越大， **ninode** 也应越大。

谁需要更改这个可调参数？

该可调参数仅与挂接了 HFS 文件系统的系统相关。请注意在 HP-UX 中不推荐使用 HFS 文件系统。

对于更改的限制

此可调参数为静态的。对该可调参数值的任何更改都需在系统重新引导后才可生效。

何时应增加此可调参数的值？

当 HFS 文件系统需要打开大量文件时，运行此应用程序的系统可能需要增大 **ninode** 的值。

增加此可调参数的副作用是什么？

由于 HFS 的 i 节点表是静态分配的，所以增大该可调参数值将消耗更多的内存。

何时应降低此可调参数的值？

在没有挂接或只有较小 HFS 文件系统的系统中可减小 **ninode** 的值，以减少内存消耗。

降低此值的副作用是什么？

内存中可保留的打开的 HFS 的 i 节点数受到这个新的较小值的限制。

同时还应更改哪些其他可调参数的值？

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。未来的 HP-UX 版本可能会删除此参数，或改变其含义。

可调参数 **ninode** 仅与挂接了 HFS 文件系统的系统相关。请注意在 HP-UX 中不推荐使用 HFS 文件系统。这个以

及其他与 HFS 相关的可调参数可能都会从将来的 HP-UX 版本中删除。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

ninode 由 HP 开发。

另请参阅

kctune(1M)、 sam(1M)、 gettune(2)、 settune(2)。

名称

nkthread - 限制允许同时运行的线程数量

值

保证安全

8416

缺省值

((nproc*2)+16) 或者 **8416**

允许值

200 - 250,000

nkthread 必须大于 **nproc + 100** 。

nkthread 必须大于 **max_thread_proc** 。

可将允许值设置得高一些，但超出部分将不被使用。

建议值

只要系统中没有大量多线程应用程序，并且 **nproc** 设置正确，缺省公式已经够用。

说明

任意给定时刻系统中允许的线程绝对数量由可调参数 **nkthread** 控制。增大它可允许更多的线程，减小它则将限制线程的数量。

如果在消息缓冲区出现 **kthread: table is full** 消息，可以确定 **nkthread** 过低。可通过 **dmesg** 或 **syslog** 来阅读这条消息。该消息表示一个应用程序不能创建线程。将 **nkthread** 值设得太低将导致应用程序因不能创建新线程或派生新进程而失败。

可通过调用 **pstat_dynamic** 并检查 **psd_numkthreadsallocd** 来确定在显示以上消息前已使用了多少同步线程。该字段表示已同步使用的线程数量的“high water”标记。

更改此可调参数的人员

任何要运行大量线程的用户。

更改限制

无。此可调参数为动态的。

应该何时增加此可调参数的值

在大多数运行了许多线程或进程的系统中，都可直接或通过调整 **nproc** 来增大该值。

增加此值的负面影响

无。

应该何时降低此可调参数的值

仅在要限制系统中的线程数量，或有内存压力，或是 **nkthread** 的值远大于预期使用值时，降低此可调参数的值。

降低此值的负面影响

增加了应用程序因不能创建新线程或派生新进程而失败的风险。

应该同时更改的其他可调参数值

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

nkthread 由 HP 开发。

另请参阅

max_thread_proc(5)、 nproc(5)。

名称

nodehostnamesize - 节点名和主机名的大小

说明

HP-UX 操作系统缺省配置允许系统节点名和主机名分别使用 8 字节和 64 字节的最大长度。节点名支持 UUCP 实用程序。主机名支持内部域名服务 (DNS) 主机名标签。

仅当启用了相应的配置选项时，才可以设置大于 8 字节的节点名，或大于 64 字节的主机名。

实际应用信息

有关详细信息，请参阅白皮书《Node and Host Name Sizes on HP-UX: Using the Expanded Capabilities》。

警告

强烈建议在设置较大的节点名或主机名之前，完全理解所有相关的文档。如果节点名大于 8 字节，或者主机名大于 64 字节，则可能导致使用 **uname** 和（或） **hostname** 命令，或使用 **uname()** 和（或） **gethostname()** 系统函数来获取名称的应用程序出现异常行为或失败。

另请参阅

hostname(1)、 uname(1)、 uucp(1)、 setuname(1M)、 gethostname(2)、 sethostname(2)、 uname(2)、 hostname(5)。

《Node and Host Name Sizes on HP-UX: Using the Expanded Capabilities》白皮书，可从 **http://docs.hp.com** 上获得。

名称

nproc - 限制允许同时存在的进程数量

值

保证安全

4200

缺省值

4200

允许值

100 - 60000

系统允许设置高于 60,000 的值，但并不能支持这样的值。将 **nproc** 设置为小于 **110** 将影响系统在多用户模式下的运行能力。某些配置可能有更高的最小值。

nkthread 必须大于 **nproc + 100**。

nproc 必须大于 **maxuprc + 5** 。

建议值

每个处理器 **1000** 个进程。

说明

可调参数 **nproc** 控制系统在任意给定时间所允许的进程的绝对数量。增大此参数可允许更多的进程，减小参数则将限制进程的数量。

如果消息缓冲区中出现 **proc: table is full** 消息，可以确定 **nproc** 过低。请使用 **dmesg** 或 **syslog** 来读取消息缓冲区。该消息表示某个应用程序不能创建一个新的进程。将 **nproc** 值设置得太低将导致应用程序因不能派生新进程而失败。

通过调用 **pstat_dynamic** 并检查 **psd_numprocsallocd** 可以确定已使用了多少个同步进程。该字段表示已同步使用的进程数量的“高水印”。

更改此可调参数的人员

任何要运行大量进程的用户。

更改限制

无。此可调参数为动态的。

应该何时增加此可调参数的值

在同时运行大量进程时，应增加 **nproc** 的值。

增加此值的负面影响

无。

应该何时降低此可调参数的值

仅当需要限制系统上的进程的数量时，或者当内存紧张且 **nproc** 值远高于预期使用值时，才应降低此可调参数的值。

降低此值的负面影响

降低此值会增加应用程序因不能派生新进程而失败的风险。

应该同时更改的其他可调参数值

增加 **nkthread** 来确保每个进程拥有足够的线程。

调整 **ksi_alloc_max**，因为缺省的公式是 **nproc**的倍数。

单个用户可能不会使用系统上的所有进程（请参阅 **maxuprc**）。

在以往，其中某些可调参数将自动调整。而现在，这些调整必须明确执行方可生效。

其他可调参数可能需要在重新引导后才生效。因此，在不重新引导的情况下大幅度增加 **nproc** 的值应谨慎进行。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

允许同时存在的进程的有效数量可能会受到可调参数 **process_id_max** 和 **process_id_min** 的影响。

某些应用程序在编码时可能已采用进程数不能超过 30,000 这个内置假定，这个进程数是之前的 HP-UX 版本强制使用的最大值。在允许和（或）创建了超过 30,000 个进程的配置上，这样的应用程序可能会失败。

HP-UX 内核将创建引导时为 **nproc** 可调参数的值优化的内部数据结构。在对此可调参数进行重大变更后，建议应有计划地重新引导，以便内核能够重新优化这些内部数据结构。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

nproc 由 HP 开发。

另请参阅

ksi_alloc_max(5)、 maxuprc(5)、 nkthread(5)、 process_id_max(5)、 process_id_min(5)。

《Number of Processes and Process ID Values on HP-UX》白皮书，可从 **http://docs.hp.com** 上获得。

名称

npty - BSD 伪终端 (pty) 的最大数量

值

缺省值

npty = **60**

允许值

最小值：**1**

最大值：系统内存大小

说明

npty 是系统可支持的伪终端 (pty) 驱动程序数。 pty 驱动程序支持被称为伪终端的一组设备对。伪终端可以是一对字符设备、一个主要设备和一个从属设备。这些设备允许一个应用程序进程和一个服务器进程间的通信。向 pty 设备（例如窗口）发送数据时，每个在任何给定时刻打开的窗口都必须存在一个 pty 设备。

不推荐使用一个远大于 pty 数的参数值。过大的参数值会浪费内核内存空间。

更改此可调参数的人员

拥有 **SYSATTR** 权限的用户。有关对支持精细划分权限的系统的授权访问的详细信息，请访问 *privileges*(5) 。

更改限制

对此可调参数的更改将在下次重新引导时生效。

应该何时增加此可调参数的值

当系统运行耗尽 pty 时应增加此可调参数值。

增加此值的负面影响

使用更多系统内存。

应该何时降低此可调参数的值

在创建一个最小化系统时应降低该值。

降低此值的负面影响

系统可能会耗尽 pty。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

npty 由 HP 开发。

另请参阅

kctune(1M)、 privileges(5)、 pty(7)。

名称

NSTREVENT - 未完成的 STREAMS bufcall 的最大数量

值

无故障

50

缺省

50

允许值

0 - 2147483647

建议值

50

说明

此可调参数限制了在任意给定的时间系统中允许存在的未完成 **bufcalls** 的最大数量。

在所有的数据流上运行的模块组合都会生成一些额外的 **bufcalls** ，从而导致资源超负荷；此可调参数可用于防止系统资源超负荷。选定的参数值应大于等于正常操作期间系统所有数据流中合理需要的 **bufcalls** 的最大值组合。在低内存情况下，STREAMS 模块使用 **bufcalls** 。

应该由谁来更改此可调参数？

任何客户。

对于更改的限制

对此可调参数的更改将在下次重新引导时生效。

何时应增加此可调参数的值？

当系统有大量低内存情况时。

增加此可调参数值的副作用是什么？

如果选择的可调参数值过大，STREAMS 子系统会向内部数据结构预分配不必要的内存。这就减少了可以提供给应用程序和系统的内存。

何时应降低此可调参数的值？

如果为某个特定 STREAMS 模块/驱动程序增加了此可调参数，删除此 STREAMS 模块/驱动程序可降低此参数的值。它应该被重新设定为原来的值。但是 HP 不推荐采用低于缺省值的取值。

降低此可调参数值的副作用是什么？

在低内存情况下，它会降低系统性能。

同时还应更改哪些其他可调参数的值？

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。未来的 HP-UX 版本可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

NSTREVENT 由 HP 开发。

名称

nstrpty - 基于 STREAMS 的伪终端 (pts) 的最大数量

值

缺省值

nstrpty = 60

允许值

最小值：**1**

最大值：系统内存大小

说明

nstrpty 是指一个系统所能支持的基于 STREAMS 的伪终端 (pts) 驱动程序的数量。 pty 驱动程序对名为伪终端的设备对提供支持。伪终端是一对字符设备，即一个主设备和一个从属设备。这就允许了在应用程序进程和服务器进程间通信。当向 pty 设备（例如 windows）发送数据时，pty 设备必须存在于任何给定的时间内打开的所有窗口中。

nstrpty 的值应当设为等于或大于 pty 设备数量，这些 pty 设备是用于基于 STREAMS 的 I/O 管道的系统中的。建议不要使用显著大于 pty 数量的参数值。 **nstrpty** 用于在内核中生成数据结构，来支持基于 STREAMS 的 pty，额外过大的取值浪费内核内存空间。

更改此可调参数的人员

拥有 **SYSATTR** 权限的用户。有关对支持精细划分权限的系统的授权访问的详细信息，请访问 *privileges*(5) 。

更改限制

对此可调参数的更改将在下次重新引导时生效。

应该何时增加此可调参数的值

当系统运行耗尽 pty 时应增加此可调参数值。

增加此值的负面影响

使用更多系统内存。

应该何时降低此可调参数的值

在创建一个最小化系统时应降低该值。

降低此值的负面影响

系统可能会耗尽 pty。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在

用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

nstrpty 由 HP 开发。

另请参阅

kctune(1M)、 privileges(5)、 pts(7)。

名称

NSTRPUSH - 单个数据流中 STREAMS 模块的最大数量

值

无故障

16

缺省

16

允许值

0 - 2147483647

建议值

16

说明

此可调参数定义了可以置入数据流的 STREAMS 模块的最大数量。这为可能会自动选取模块并置入某一数据流中的异常进程提供了保护。这样做的意图并不是为了作为一种防护措施来限制系统用户对于 STREAMS 模块的恶意使用。

在一个数据流中，大多数系统需要的模块不超过三到四个。然而，也可能有需要更多模块的不普遍情况。这一可调参数的缺省值允许在一个数据流中设置多达 16 个模块，这一设定应该可以满足哪怕是最苛刻的安装和应用程序的需求。

应该由谁来更改此可调参数?

任何客户。

对于更改的限制

对此可调参数的更改将在下次重新引导时生效。

何时应增加此可调参数的值?

当客户需要在单一数据流中置入更多的 STREAMS 模块时。

增加此可调参数值的副作用是什么?

可能导致异常应用程序过度消耗系统资源。

何时应降低此可调参数的值?

没有必要将可调参数值从缺省值调低。

降低此可调参数值的副作用是什么?

太低的取值可能会导致网络命令失败。

同时还应更改哪些其他可调参数的值?

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。未来的 HP-UX 版本可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

NSTRPUSH 由 HP 开发。

名称

NSTRSCHED - 能运行的 STREAMS 调度守护程序的数量

值

无故障

0

缺省

0

允许值

0 - 2147483647

建议值

0

说明

此可调参数定义了在超过一个处理器的系统上要运行的多处理器 (MP) STREAMS 调度守护程序的数量。请注意，单处理器 (UP) 系统不使用 MP 调度守护程序，但 MP 和 UP 系统两者都拥有一个 UP STREAMS 调度程序 (supsched)。

如果该可调参数值设定为零，则由系统根据系统中处理器数量确定使用多少守护程序。如果此可调参数的取值设定为一个非零正值，则该数值为在多处理器系统中将建立的 smpsched 守护程序的数量。

注释：此可调参数仅用于特定的 HP 产品。可能在未来的 HP-UX 版本中删除。

应该由谁来更改此可调参数？

任何客户。

对于更改的限制

对此可调参数的更改将在下次重新引导时生效。

何时应增加此可调参数的值？

此可调参数仅用于特定的 HP 产品。可能在未来的 HP-UX 版本中删除。

增加此可调参数值的副作用是什么？

可能使系统性能的更改不可预测。

何时应降低此可调参数的值？

此可调参数仅用于特定的 HP 产品。可能在未来的 HP-UX 版本中删除。

降低此可调参数值的副作用是什么？

可能使系统性能的更改不可预测。

同时还应更改哪些其他可调参数的值？

无。

警告

此可调参数仅用于特定的 HP 产品。

所有 HP-UX 内核可调参数都是针对于特定发行版的。未来的 HP-UX 版本可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

NSTRSCHED 由 HP 开发。

名称

nstrtel - 指定内核可支持传入 telnet 会话的 telnet 设备文件的数量

值

无故障

60

缺省

60

允许值

Any positive integer. （依据可用物理内存而定）。

最好使用缺省值，并且不需要降低此缺省的参数值。然而，如果同时的 telnet 连接负荷过高，那么 **nstrtel** 的值可以增加。

建议值

60 （缺省值）。

说明

telnet 守护程序使用两个基于 STREAMS 的伪终端驱动程序（ telm 和 **tels** ）。内核参数 **nstrtel** 可以用于调整伪终端数量。 **nstrtel** 指定系统引导时创建的内核数据结构的数量，这些数据结构是支持服务器上传入 telnet 会话所需要使用的设备文件。如果使用 **insf** 命令或者 SAM 来创建更多的 telnet 设备文件，相应的 **nstrtel** 的取值就必须增加，否则，会因没有与系统通讯的内核数据结构而导致设备文件不可用。

应该由谁来更改此可调参数?

任何人。

对于更改的限制

对此可调参数的更改将在下次重新引导时生效。

何时应增加此可调参数的值?

当 telnet 连接负荷上升而且没有打开 telnet 连接的设备文件可用时，应该调高这一可调参数的值。

当发生 **telnetd: Telnet device drivers missing: No such device** 错误时，说明设备文件已经耗尽，应该调高 **nstrtel** 的值。调高 **nstrtel** 的取值后，应该运行 **insf** 来创建新的设备文件。（如果使用 SAM 调高 **nstrtel** 的值，则系统会自动运行 **insf** ）。

增加此可调参数值的副作用是什么?

会消耗更多资源。额外的内核数据结构和额外的设备文件会阻塞系统。

何时应降低此可调参数的值?

不要使可调参数值的设置低于缺省值。如果要设置此可调参数低于缺省值，应先咨询 HP 技术支持。

降低此可调参数值的副作用是什么?

虽然不推荐，但是并没有任何副作用。

同时还应更改哪些其他可调参数的值?

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。未来的 HP-UX 版本可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

nstrtel 由 HP 开发。

另请参阅

insf(1M)、 telnetd(1M)、 telm(7)、 tels(7)。

名称

nswapdev - 可用于交换的设备的最大数量

值

缺省值

32 个设备

允许值

最小值： **1** 个设备

最大值： **1024** 个设备

说明

为了索引更方便，交换设备在内核中是用表格管理的。 **nswapdev** 设定了此表中内核变量的上限，进而也就设定了可以用于交换的设备的上限。

更改此可调参数的人员

任何用户。

更改限制

对此可调参数的更改将在下次重新引导时生效。

应该何时增加此可调参数的值

如果系统中添加了另外一个交换设备，则需要在 **nswapdev** 的基础上增加设备数，同时 **swapon()** 返回 [ENOENT] 给调用方（请参阅 *swapon*(2) 联机帮助页）。

增加此值的负面影响

可以给系统添加更多的设备，因此内核将需要多一点内存来保存表格。一个不严重的性能方面的负面影响是内核要在 **swapon** 也为 true 时为一个复制设备而扫描更多的设备，但实际上这种情况可以忽略。

应该何时降低此可调参数的值

仅当能确定系统永远不会使用超过某一特定数量的交换设备时，并且希望通过执行 **swapon** 操作降低这一可调参数来节省少量的内核内存，提高内核性能。

降低此值的负面影响

除了主要预计会出现的交换设备数量的新的限制，没有其他负面影响。

应该同时更改的其他可调参数值

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

nswapdev 由 HP 开发。

名称

nswapfs - 可用于交换的文件系统的最大数量

值

缺省值

32 个文件系统

允许值

最小值： **0** 个文件系统

最大值： **1024** 个文件系统

说明

为了索引更方便，文件系统交换设备在内核中是用表格管理的。 **nswapfs** 设定了此表中内核变量的上限，进而也就设定了可以用于交换的文件系统的上限。

更改此可调参数的人员

任何用户。

更改限制

对此可调参数的更改将在下次重新引导时生效。

应该何时增加此可调参数的值

如果系统中添加了另外一个文件系统交换，则会使数量超过 **nswapfs** ，此时 **swapon()** 返回 [ENOENT] 给调用方（请参阅 *swapon*(2) 联机帮助页）。

增加此值的负面影响

系统中可以添加更多的用于交换的文件系统，因此内核需要一点额外的内存来保存表格。增加此值产生的一个不严重的性能方面的负面影响是在 swapon 过程中，内核必须扫描更多的文件系统以检查一个文件系统副本，但实际上可以忽略这个负面影响。

应该何时降低此可调参数的值

仅当能确定系统永远不会使用超过某一特定数量的交换文件系统时，并且希望通过执行 swapon 操作，降低这一可调参数来节省少量的内核内存，提高内核性能。

降低此值的负面影响

除了主要预计会出现的交换文件系统数量的新的限制，没有其他负面影响。

应该同时更改的其他可调参数值

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在

用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

nswapfs 由 HP 开发。

名称

nsysmap、nsysmap64 - 内核动态内存分配映射中的条目数

值

缺省值

8400

允许值

任意正值。

说明

注释：在 HP-UX 11i v3 中，将不再出现这些可调参数。系统将为表建立一个初始的大小，并根据需要增加其大小。在将来的版本中将删除此联机帮助页。

预发行的 11i v3 的用法

此值设定了内核动态内存资源映射的大小；该映射是一个地址/长度对的数组，它描述了内核的动态地址空间中的可用虚拟空间。

此数组的大小在过去是静态的。这意味着特定不合理的工作负荷会让内核地址空间变得非常零碎，从而在数组中产生过多条目。发生这种情况时，系统不会出现混乱，而是会抛弃最后的条目，结果导致出现“泄漏”的内核虚拟地址空间。如果这种溢出发生得过于频繁，系统会逐渐耗尽虚拟空间，进而出现混乱并显示以下消息：

kalloc: out of virtual space.

通过让映射的大小可调，系统可以根据系统工作负荷的大小自动调整映射的大小，从而避免出现此问题。如果自动调整不起作用，可以手动进行调整，使之与特定的工作负荷相适应。当缺省值被覆盖时，内核可能会增加该值，使之超出规范，具体取决于系统的大小。

针对 32 位和 64 位内核有不同的可调参数，因为 64 位内核具有更多虚拟地址空间。 **nsysmap** 可调参数控制着 32 位内核， **nsysmap64** 可调参数控制着 64 位内核。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

nsysmap 和 **nsysmap64** 由 HP 开发。

名称

numa_policy - 基于单元的 HP-UX 服务器上的物理内存分配策略

值

缺省值

0

允许值

最小值： **0**

最大值： **2**

说明

大型服务器是由一个或多个称为单元的组成单元构建而成的。每个单元至少有 1 个 CPU，并且通常有一定数量的内存。这些基于单元的服务器使用非一致性内存访问 (NUMA) 协议。这意味着应用程序线程可能会遇到差异较大的内存滞后时间，这取决于该线程是从其运行所在的同一单元中访问内存，还是从不同的单元访问内存。

HP-UX 执行许多内存分配功能来优化性能。用户可以将系统上的内存划分为两类： “交叉内存”和 “单元本地内存”。

交叉内存提供一致的滞后时间。它使 NUMA 硬件如同单个一致内存组，从而为所有应用程序在所有 CPU 间提供相同的平均滞后时间。在整个内存构造中分配内存负荷时，这种功能非常有用。缺省情况下，将 HP-UX 服务器中的所有内存配置为交叉内存。

对于在其访问的内存所在的单元中运行的进程，基于单元的内存为该进程提供较短的滞后时间（高性能） 。相反，来自远程单元的引用具有的滞后时间要长得多。

用户可以使用 HP-UX 命令 *parmodify*(1M) 配置单元本地内存。

用户可以使用 **numa_policy** 动态内核可调参数定义如何利用交叉内存和单元本地内存。有三个可能值：

numa_policy = 0

这是缺省策略。在既有单元本地内存也有交叉内存的计算机中， HP-UX 可以就如何通过区分专用内存和共享内存来使用内存，做出明智的决策。分配给专用对象（堆栈、堆、专用文件等）的内存很可能来自单元本地内存。除非用户另行指定对文件使用 *fadvise*(2) ，或者使用 *shmget*(2) 中的内存选项之一，否则，分配给共享对象（共享文件、System V 共享内存）的内存都是从交叉内存中分配的。

numa_policy = 1

此策略指示 HP-UX 对所有内存分配使用单元本地内存。此策略将选择要分配其中的位置的单元，分配的方式是首先遵循 **fadvise()** 或 **shmget()** 中指定的任何单元位置策略。如果未指定单元位置策略，则此策略会从本地单元中分配所有内存。如果本地单元中的内存不足，则使用最靠近的单元中的内存，依此类推。即使使用了交叉内存，也仅仅是因为可用的单元本地内存距离太远。如果使用此选项，则需要注意确保在分配了内存的同一单元中调度应用程序。有关详细信息，请参阅 *mpsched*(1) 。

numa_policy = 2

此策略指示 HP-UX 覆盖通过 **fadvise()** 或 **shmget()** 提供的任何位置策略，并从交叉内存中分配所有内存。如果交叉内存中的内存不足，则使用单元本地内存。

更改限制

对此可调参数所做的更改，将作用于在更改以后分配的所有内存。在更改之前创建的任何内存均不受影响。

何时应将此可调参数的值更改为 1

仅当系统上的所有性能敏感应用程序都具有密度较高的单元位置时，才应该将此可调参数值设置为 1。也就是说，如果对共享内存的大多数引用都是来自在分配了内存的那一个单元上运行的进程，则 **numa_policy = 2** 设置将对性能产生正面影响。完成共享分配后，应该将此值更改回 0。使用 *parmodify*(1M) 将足够的内存指定为单元本地内存，并且重新引导计算机后，应该将值更改为 1。

何时应将此可调参数的值更改为 2

如果工作负荷具有密度较低的单元位置，并且无法更改源代码，则可以将此可调参数值设置为 2。完成内存分配后，应该将此值更改为 0。使用 *parmodify*(1M) 将足够的内存指定为交叉内存，并且重新引导计算机后，应该将值更改为 2。请注意，将 **numa_policy** 可调参数设置为 2 仅在这样的混合型环境中起作用：其中一些应用程序能够受益于单元本地内存，而另一些却不能。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

numa_policy 由 HP 开发。

另请参阅

mpsched(1)、 parmodify(1M)、 fadvise(2)、shmget(2)。

名称

orientation - 数据流的方向

说明

数据流的方向是一个作为输入/输出流处理的 **FILE** 对象的属性。输入/输出模式假定一个应用程序中的字符按照宽字符处理，并按照多字节字符保存于文件中，还假定所有宽字符输入/输出函数都与定位在两个多字节字符之间的边界上的数据流一起开始执行，这是非常有用的功能。

在一个数据流与一个文件关联后但尚未在数据流上执行任何操作前，数据流是没有方向的。如果一个宽字符输入或输出函数应用于一个无方向的数据流，那么该数据流将隐式成为一个面向宽字符的数据流。同样的，如果一个字节输入或者输出操作应用于一个无方向的数据流，那么该数据流将隐式成为一个面向字节的数据流。当数据流没有方向时，只有 **fwide()** 函数可以显式地更改一个数据流的方向。

只有当数据流由 **popen()** 函数关联到管道后，数据流才是面向字节的。

在数据流成为面向字节的或者面向宽字符的数据流后，数据流的方向将固定下来，且不可更改，直到数据流关闭。

下列函数是宽字符输入/输出函数。

> **fgetwc()** 、 **fgetws()** 、 **fputwc()** 、 **fputws()** 、 **fwprintf()** 、 **fwscanf()** 、 **getwc()** 、 **getwchar()** 、 **putwc()** 、 **putwchar()** 、 **putws()** 、 **ungetwc()** 、 **vfwprintf()** 、 **wprintf()** 、 **wscanf()** 。

下列函数是字节输入/输出函数。

> **fgetc()** 、 **fgets()** 、 **fprintf()** 、 **fputc()** 、 **fputs()** 、 **fread()** 、 **fscanf()** 、 **fwrite()** 、 **getc()** 、 **getchar()**、 **getc_unlocked()** 、 **getchar_unlocked()** 、 **gets()** 、 **getw()** 、 **printf()** 、 **putc()** 、 **putchar()** 、 **putc_unlocked()** 、 **putchar_unlocked()** 、 **puts()** 、 **putw()** 、 **scanf()** 、 **ungetc()** 、 **vfprintf()** 、 **vprintf()** 。

举例

当数据流方向未知时从数据流中读取字符：

```
int     so;
wchar_t ws[CHAR_NUM];
char    s[CHAR_NUM];

so = fwide(stream, 0); /* Check the orientation */
if (so > 0) /* the stream is wide-oriented */
   fgetws(ws, CHAR_NUM, stream);
else if (so < 0) / * stream is byte-oriented */
   fgets(s, CHAR_NUM, stream);
else /* the stream is without orientation */
   fprintf(stderr, "It is first time to access this file.");
```

警告

如果将字节输入/输出函数应用于面向宽字符的数据流，或将宽字符输入/输出函数应用于面向字节的数据流，会导致行为未定义的后果。

作者

数据流方向的功能由 HP 和 Mitsubishi Electric Corporation 联合开发。

另请参阅

fgetws(3C)、 fopen(3S)、 fread(3S)、 fwide(3C)、 fwprintf(3C)、 fwscanf(3C)、 getc(3S)、 gets(3S)、 getwc(3C)、 popen(3S)、 printf(3S)、 putc(3S)、 puts(3S)、 putwc(3C)、 putws(3C)、 scanf(3S)、 ungetc(3S)、 ungetwc(3C)。

名称

pagezero_daemon_enabled - 启用后台中的可用内存的清零

值

缺省

1

允许值

最小值： **0**

最大值： **1**

说明

HP-UX 通过将分配给用户空间的任意内存清零改善了安全性。这确保了任何用户都不能读取其他用户写入的内容。一般情况下，清零是在将物理页分配给用户时进行的，并且通常是在应用程序首次触及该页之时。某些系统调用，像 **mlock()** ，也会导致页面的清零。这类系统调用和访问所用的时间取决于被清零的内存的大小。一个 4G 的页面可在若干秒内轻松分配完毕。一个大型数据库的共享内存段的分配可能需要若干分钟。而较小页面的分配通常感觉不到。

pagezero 守护程序旨在增强性能，它可以减少完成诸如页面错误、 **mlock()** 等内核操作所需的时间。其理念是在 CPU 空闲时，将较大的可用页面（4MB 及以上）清零。该守护程序是专门设计的，可确保在 CPU 空闲时仅执行短暂的时间。

但是，在特定情况下，如果某些资源（CPU、TLB 或内存带宽）的利用率非常高，则该守护程序的运行反而会影响性能。例如，如果应用程序受到内存带宽的限制，则禁用该守护程序可能会更好。这种情况比较少见。大多数工作负荷都不需要禁用该守护程序。

此可调参数允许系统管理员禁用和启用 **pagezero** 守护程序。当该守护程序被禁用时，它将不会对任何页面进行清零。进程中已经被清零的任何页面将被清零。当启用该守护程序时，它会将任何大小为 4MB 及以上的未清零的可用页面清零。

应该由谁更改此可调参数？

任何人。

对于更改的限制

此可调参数的更改立即生效。

何时应将此可调参数的值更改为 **0**？

仅当系统使用了较大的页面（即 **vps_ceiling** 被设置为 4MB 或以上）时才应更改此可调参数的值。如果系统使用的页面大小不超过 4M，更改此值并没有什么作用。如果系统在某些硬件资源上出现瓶颈，则可以将可调参数更改为 0。尤其是禁用该守护程序将有助于提高 TLB 命中率、增加 CPU 的可用时间以及减少内存的滞后现象。

将此值更改为 **0** 的副作用是什么？

禁用该守护程序会增加处理页面错误和完成系统调用的时间，这两者都会导致内存的分配（例如， **mlock()** ）。

警告

所有 HP-UX 内核可调参数都特定于各个发行版本。未来的 HP-UX 版本可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

pagezero_daemon_enabled 由 HP 开发。

另请参阅

vps_ceiling(5)。

名称

pam_authz - 用于提供用户授权的 PAM 模块

概要

/usr/lib/security/$ISA/libpam_authz.so.1

说明

管理员可通过 PAM **/usr/lib/security/$ISA/libpam_authz.so.1** 的 **pam_authz** 服务模块提供这样的功能：根据 **/etc/passwd** 文件中显示的 **netgroup** 信息或访问策略文件 **/etc/opt/ldapux/pam_authz.policy** 中定义的访问规则控制登录系统的用户。

缺省情况下，创建 **pam_authz** 是为了提供类似于 NIS 执行的 **netgroup** 过滤功能的访问控制。不使用 NIS 时（例如，使用了 **pam_ldap** 或 **pam_kerberos** 身份验证模块时），将使用 **pam_authz** 。由于 **pam_authz** 不提供身份验证，因此它不会验证某个用户帐户是否存在。

pam_authz 还将其功能扩展为定义主机和服务访问管理策略。 **pam_authz** 支持本地访问策略文件，您可以通过此文件根据各种信息定义访问规则。可根据 LDAP X.500 样式组、常规 POSIX 组、 **netgroups** 、 **ldap** 过滤器和各个用户定义 **allow** 或 **deny** 访问规则。要激活此功能，请在 **/etc/opt/ldapux** 下面创建 **pam_authz.policy** 文件。

pam_authz 为下列所有四个 PAM 组件提供接口：身份验证、帐户管理、会话管理和口令管理。但是，只需要配置帐户管理组件。用于会话管理和口令管理的 PAM 组件为 NULL 函数。这些组件始终返回 [PAM_SUCCESS] 。

libpam_authz.so.1 库是一个共享对象，可以根据需要动态加载以提供所需的功能。其路径在 PAM 配置文件中指定。

身份验证和帐户管理模块

pam_authz 身份验证组件不提供身份验证。但它通过 **pam_sm_acct_mgmt()** 提供授权。 **pam_authz** 专门与其他身份验证模块一起用作补充模块，其中，另一个模块用于验证用户身份，而 **pam_authz** 用于验证用户访问权限。如果有权限访问系统的用户的列表是大型储备库（例如 LDAP 目录服务器或其他数据库）中存储的用户的子集，则使用 **pam_authz** 。

由于 **pam_authz** 只提供授权而不提供身份验证，因此，强烈建议在配置文件中将 **pam_authz** 设置为 **required** （请参阅 *pam.conf*(4) ）。通常，在 **/etc/pam.conf** 文件的帐户管理区段中将 **pam_authz** 配置为 **first** 模块。

但是，对于不调用 PAM 帐户管理过程的 PAM 应用程序，还可将 **pam_authz** 配置为身份验证模块。如果将 **pam_authz** 配置为身份验证模块，则必须至少将其他一个 PAM 模块设置为 **required** 才可以验证用户。

如果不存在访问策略文件 **/etc/opt/ldapux/pam_authz.policy** ， **pam_sm_acct_mgmt()** 将使用 **netgroups** （请参阅 *netgroup*(4) ）和 **/etc/passwd** 文件确定用户访问权限，并使用与 NIS 定义的语法类似的语法。但是， **pam_authz** 不支持 NIS 定义的口令条目过滤语法，而是根据口令字段是否阻塞来确定是否应向某个 **netgroup** 成员授予（或拒绝）访问权限。

pam_authz 将扫描 **/etc/passwd** 文件以查找匹配的 NIS 样式条目，并根据与涉及的帐户相匹配的第一个规则，返回授予或拒绝访问权限的结果。例如，如果在 **/etc/passwd** 文件中定义了下列条目，则 **pam_authz** 将授予或拒绝访问权限：

+ 向数据库中的所有用户授予访问权限。

+@*name* 向网络组 *name* 的所有成员授予访问权限。

+*name* 向用户 *name* 授予访问权限。

+@*name***:***any_non_NULL_string*
拒绝向网络组 *name* 的所有成员授予访问权限。

+*name***:*** 拒绝向用户 *name* 授予访问权限。

-@*name* 拒绝向网络组 *name* 的所有成员授予访问权限。

-*name* 拒绝向用户 *name* 授予访问权限。

有关 **/etc/passwd** 示例文件，请参考 *passwd*(4) 。

通过访问策略文件， **pam_sm_acct_mgmt()** 将使用 **/etc/opt/ldapux/pam_authz.policy** 文件来帮助确定哪些用户可以登录。找到授权规则之前，将评估访问策略文件中的每个访问规则。授权规则是与用户登录名匹配的第一个访问规则。 **pam_sm_acct_mgmt()** 根据授权规则的信息返回 **allow** 或 **deny** 访问权限的结果。如果找不到授权规则，系统将拒绝用户登录。

访问规则是访问策略的基本元素。“策略”是按给定顺序排列的不同访问规则集的集合。一个访问规则包括三个字段。

*action***:***type***:***object*

其中每个字段的含义如下：

action *action* 字段定义在访问规则评估结果为真的情况下授予的访问权限。该字段有两个可能值：

allow 授予登录授权

deny 限制登录授权

type *type* 字段中的值代表信息源。该值表示 **PAM_AUTHZ** 应该查找的用户信息类型。该值还有助于确定下面的 *object* 字段中的正确语法。支持下列值：

type 用法

unix_user 通过将用户登录名与 *object* 字段中的用户名列表进行比较来控制访问权限。

unix_group 通过检查用户的 posix 组成员关系来控制访问权限。 Unix POSIX 组列表在 *object* 字段中指定。 **pam_authz** 通过查询 **nsswitch.conf** 中指定的名称服务来检索列出的每个组的信息。

netgroup 通过检查用户的 **netgroup** 成员关系来控制访问权限。 **netgroup** 名称列表在 *object* 字段中指定。 **pam_authz** 通过查询 **nsswitch.conf** 中指定的名称服务来获取 **netgroup** 信息。

ldapgroup 通过检查用户的非 posix 组成员关系来控制访问权限。 **pam_authz** 支持使用 **groupOfNames** 或 **groupOfUniqueNames** 对象类的 X.500 样式组。 **pam_authz** 通过 LDAP-UX 客户端从目录服务器检索列出的每个组的成员关系。

ldapfilter 通过检查组织中的用户角色来控制访问权限。 **pam_authz** 使用提供的 ldap 过滤器查询用户 ldap 信息。

other **other** 访问规则用作通配符规则。使用此规则 允许或 拒绝对所有用户的访问权限。

object *object* 字段中的值定义使用登录名验证 **pam_authz** 时需要遵循的条件。下表提供 *object* 字段的所有可能值和语法的摘要。

type	**object**
unix_user	该字段包含用户名列表。每个值（用户名）都是以逗号 (**,**) 分隔符分隔的字符串，格式为 ASCII 2C HEX。该字段为多值字段。
unix_group	该字段包含 unix 组名列表。每个值（组名）都是以逗号 (**,**) 分隔符分隔的字符串，格式为 ASCII 2C HEX。该字段为多值字段。
netgroup	该字段包含 **netgroup** 名称列表。每个值（组名）都是以逗号分隔符 (**,**) 分隔的字符串，格式为 ASCII 2C HEX。该字段为多值字段。
ldapgroup	该字段包含使用 **groupOfNames** 对象类或 **groupOfUniqueNames** 对象类的 LDAP 组（非 Posix 组）的判别名 (DN)。 ND 语法在 RFC2253 中定义。该字段为单值字段。不需要使用分隔符。只允许使用一个判别名。 在 **ldapfilter** 访问规则中，该字段包含单个搜索过滤器，用于指定一个或多个 (**attribute=***value*) 对。字符串搜索过滤器的语法在 RFC2254 单值字段中定义。不需要使用分隔符。仅允许使用一个搜索过滤器。

下面是 **/etc/opt/ldapux/pam_authz.policy** 中的访问规则的示例：

```
allow:unix_user:peter,john,mary
allow:unix_group:admin,operator,support
deny:unix_group:guest,contractor,vendor
allow:netgroup:netcom,netprint,netmail
allow:ldap_group:cn=admingroup,ou=eng,dc=example,dc=com
allow:ldap_filter:(&(manager=tomc)(departmentnumber=113))
```

可将下列选项传递给 **pam_authz** 服务模块：

调试 **syslog() LOG_DEBUG** 级别的调试信息。

nowarn 关闭警告消息。

use_first_pass 将忽略该选项。

try_first_pass 将忽略该选项。

pam_sm_setcred() 函数可设置特定于用户的凭据。对于 pam_authz，这是一个 NULL 函数。

会话管理模块

UNIX 会话管理组件提供了相应函数，用于启动 (**pam_sm_open_session()**) 和终止 (**pam_sm_close_session()**) 会话。对于 **pam_authz** ， **pam_open_session()** 是一个 NULL 函数。可将下列选项传入 **pam_authz** 服务模块：

调试 **syslog() LOG_DEBUG** 级别的调试信息。

nowarn 关闭警告消息。

pam_close_session 是一个 NULL 函数。

口令管理模块

口令管理组件提供了可用于更改口令的函数 (**pam_sm_chauthtok()**) 。对于 **pam_authz** ，该模块是一个 NULL 函数。可将下列选项传入 pam_authz 服务模块：

调试 **syslog() LOG_DEBUG** 级别的调试信息。

nowarn 关闭警告消息。

use_first_pass 将忽略该选项。

try_first_pass 将忽略该选项。

举例

下面是 **pam.conf** 配置文件的示例。以 **#** 符号开头的行被视为注释，因此被忽略。

```
#
# PAM configuration
#
# Authentication management
#
login auth    required    libpam_hpsec.so.1
login auth    sufficient  libpam_unix.so.1
login auth    required    libpam_ldap.so.1 try_first_pass
OTHER auth    required    libpam_hpsec.so.1
OTHER auth    sufficient  libpam_unix.so.1
OTHER auth    required    libpam_ldap.so.1 try_first_pass
#
# Account management
#
login account required    libpam_hpsec.so.1
login account required    libpam_authz.so.1
```

```
login account sufficient  libpam_unix.so.1
login account required    libpam_ldap.so.1
OTHER account required    libpam_authz.so.1
OTHER account required    libpam_hpsec.so.1
OTHER account sufficient  libpam_unix.so.1
OTHER account required    libpam_ldap.so.1
#
# Session management
#
login session required    libpam_hpsec.so.1
login session sufficient  libpam_unix.so.1
login session required    libpam_ldap.so.1
OTHER session required    libpam_hpsec.so.1
OTHER session sufficient  libpam_unix.so.1
OTHER session required    libpam_ldap.so.1
#
# Password management
#
login password required   libpam_hpsec.so.1
login password sufficient libpam_unix.so.1
login password required   libpam_ldap.so.1 try_first_pass
OTHER password required   libpam_hpsec.so.1
OTHER password sufficient libpam_unix.so.1
OTHER password required   libpam_ldap.so.1 try_first_pass
```

另请参阅

pam(3)、 pam_authenticate(3)、 pam_setcred(3)、 syslog(3)、 netgroup(4)、 pam.conf(4)、 pam_user.conf(4)、 passwd(4)、 ldapux(5)、 pam_krb5(5)、 pam_ldap(5)。

名称

pa_maxssiz： pa_maxssiz_32bit、 pa_maxssiz_64bit - 在 Integrity 系统上，在 PA-RISC 仿真器下运行的用户进程的最大堆栈大小（以字节为单位）

值

缺省值

32 位缺省值：**0x04FC6000 (79Mb)**

64 位缺省值：**0x20000000 (512Mb)**

允许值

32 位最小值：**0x04FC6000 (79Mb)**

32 位最大值：**0x1BF00000 (447Mb)**

64 位最小值：**0x20000000 (512Mb)**

64 位最大值：**0xA0000000 (2.5Gb)**

说明

HP-UX 系统上的用户程序由五个不连续的虚拟内存段组成：文本（或代码）、数据、堆栈、共享和 I/O。

在基于 Itanium 的系统下使用 PA-RISC 仿真器运行的 PA-RISC 二进制程序通过 **pa_maxssiz_32bit** 或 **pa_maxssiz_64bit** 确定其堆栈段的大小（包括内存堆栈和 RSE 堆栈）。

如果进程是在一台没有仿真器的 PA-RISC 计算机上执行的，那么从进程的角度看，仿真堆栈与堆栈是相同的。某个仿真进程的实际（非仿真）进程堆栈（包括内存堆栈和 RSE 堆栈）的大小是不可配置的，并且仅供基于 Itanium 的系统上的 PA-RISC 进程的仿真器内部使用。因此，此可调参数代表了在相应 PA-RISC 计算机上执行仿真进程所需的 **maxssiz** 或 **maxssiz_64bit** 的值，加上仿真器本身的工作空间。

对于由 **maxssiz** 设定的堆栈地址空间大小，仿真器所使用的地址空间是专用于该进程的，并且不能用于其他目的，而不管仿真器堆栈是否增加到覆盖了整个范围。因此，该可调参数值可被视为是对此 PA-RISC 进程不可用的进程数据地址空间的量。对于 32 位进程尤其需要注意，因为缺省的地址空间模型仅提供了 1GB 的专用数据空间。将此可调参数设置得过高会导致执行失败，此时其他分配（专用的 **mmap()** 调用或 **sbrk()** 调用）会由于可用地址空间不足而失败。如果发生这种情况，请使用 **chatr** 修订失败的 PA-RISC 可执行程序的地址空间模型，以允许更多的专用数据虚拟地址空间或降低可调参数。

更改此可调参数的人员

任何用户。

更改限制

对此可调参数的更改仅在重新引导系统后才会生效。

由于此可调参数不能修改用户内存堆栈的大小，因此在修改此可调参数之前，最好对 PA-RISC 仿真器内存的使用情况有着比较详细的了解。

应该何时增加此可调参数的值

在基于 Itanium 的系统上，如果 PA-RISC 仿真器的堆栈仿真需要更多空间，则应增加此可调参数的值。

增加此值的负面影响

PA-RISC 仿真器的堆栈将使用更多的虚拟地址空间，从而减少了留给数据分配的虚拟地址空间。此值增加过高会导致应用程序失败，此时数据分配会由于虚拟地址空间不足而失败。

由于可能不知道哪个进程是基于 PA-RISC 二进制程序的，哪个是基于 Itanium 二进制程序的，因此不主张随意增加此可调参数的值。

应该何时降低此可调参数的值

仅当在系统上 PA-RISC 仿真器的工作负荷已知，并且当前用于堆栈仿真的空间比实际需要的多，而系统上的交换空间（保留的或分配的空间）又非常稀缺时，才能降低此可调参数的值。

降低此值的负面影响

降低此可调参数的值会降低仿真 PA-RISC 二进制可执行程序的可用堆栈空间。需要较大堆栈的仿真进程的堆栈增加请求会失败（因为降低 **maxssiz** 或 **maxssiz_64bit** 可能会由于无法增加堆栈而导致进程终止）。

应该同时更改的其他可调参数值

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

pa_maxssiz 由 HP 开发。

另请参阅

chatr(1)、 mmap(2)、 sbrk(2)、 maxssiz(5)。

名称

pam_hpsec - 扩展的验证、帐户、口令和会话服务 HP-UX 的模块

概要

/usr/lib/security/$ISA/libpam_hpsec.so.1

说明

hpsec 服务模块实现 HP-UX 特定的扩展，用于验证、帐户管理、口令管理和会话管理。

建议所有服务都使用 **pam_hpsec** ，但要求 **login** 、 **dtlogin** 、 **ftp** 、 **su** 、 **remsh/rexec** 和 **ssh** 等服务必须使用它。应用程序编写者和系统管理员可以判定某些特定应用程序并不适合使用 **pam_hpsec** 。当 **pam_hpsec** 模块出现在堆栈中时，它必须位于堆栈的顶部，位于 **pam_unix** 、 **pam_krb5** 或 **pam_ldap** 等其他模块的上方。本模块专用于 HP-UX，其功能在不同版本之中可能会有明显的不同。

有关模块路径的解释，请参考 *pam.conf*(4) 中的相关信息。

选项

可以将下列选项传递给所有组件的 **hpsec** 服务模块：

debug 对位于 **LOG_DEBUG** 的调试信息执行 *syslog*(3C)

操作。

nowarn 关闭警告消息。

opaque 使用本选项，成功后 **pam_hpsec** 将返回 **PAM_SUCCESS** 。如果没有本选项，成功后模块将返回 **PAM_IGNORE** （这简化了 PAM 配置）。

验证组件

hpsec 验证组件提供特定于 HP-UX 的凭证管理。将来，除了实现凭证管理之外，该组件还可能实现附加的 HP-UX 特定验证限制。

目前，此组件将初始化会话的审核属性。除了“选项”一节中列出的选项之外，还可以将下列选项传递给验证模块。

bypass_setaud 使用此选项， **pam_hpsec** 不初始化会话的审核属性。支持此选项的唯一目的在于，通过 *su*(1)配置 **pam_hpsec** 时，能够保持 *su*(1)

向后兼容的行为。建议不要将此选项用于其他服务。

bypass_all 使用此选项， **pam_hpsec** 可忽略该模块将强制执行的限制或功能。

请注意，PAM 模块并不管理其他的常用 UNIX 凭证（如 **uid** 和 **gid** ）以及补充的组成员关系。进行验证的应用程序应使用 *setuid*(2) 和 *initgroups*(3C) 类型的调用来授予这些凭证（在调用 *pam_open_session*(3) 后必须授予这些凭证）。

帐户管理组件

此组件实行了在 *security*(4) 中所述的 **AUTH_MAXTRIES** 和 **LOGIN_TIMES**

限制。除了“选项”一节中列出的选项之外，还可以将下列选项传递给帐户管理模块。

bypass_maxtries 使用此选项， **pam_hpsec** 将忽略 **AUTH_MAXTRIES** 限制。

bypass_login_times 使用此选项， **pam_hpsec** 将忽略 **LOGIN_TIMES** 限制。

bypass_all 使用此选项， **pam_hpsec** 可忽略该模块将强制执行的限制或功能。

口令管理组件

此组件无条件地运行成功。

会话管理组件

此组件实行了 *security*(4) 中所述的多种限制，如 **DISPLAY_LAST_LOGIN**、 **NOLOGIN**、 **NUMBER_OF_LOGINS_ALLOWED** 和 **UMASK**

限制。除了“选项”一节中列出的选项之外，还可以将下列选项传递给会话管理模块。

bypass_nologin 使用此选项， **pam_hpsec** 将忽略 **NOLOGIN** 设置。

bypass_limit_login 使用此选项， **pam_hpsec** 将忽略 **NUMBER_OF_LOGINS_ALLOWED** 设置。

bypass_umask 使用此选项， **pam_hpsec** 将忽略 **UMASK** 设置。

bypass_last_login 使用此选项， **pam_hpsec** 将忽略 **DISPLAY_LAST_LOGIN** 设置。

bypass_all 使用此选项， **pam_hpsec** 可忽略该模块将强制执行的限制或功能。

举例

以下是一个使用 **pam_hpsec** 模块的堆栈示例：

```
login session required    pam_hpsec.so.1
login session sufficient  pam_unix.so.1
login session sufficient  pam_ldap.so.1
login session sufficient  pam_krb5.so.1
```

以上规则表明，除 **hpsec** 外，登录的会话管理至少需要 UNIX、LDAP 和 Kerberos PAM 模块中任意一个模块。

作者

pam_hpsec 由 HP 开发。

另请参阅

pam(3)、 pam_acct_mgmt(3)、 pam_authenticate(3)、 pam_open_session(3)、 pam_setcred(3)、 pam.conf(4)、 security(4)、 userdb(4)。

名称

pam_ldap - LDAP 的身份验证、帐户、会话和口令管理 PAM 模块

概要

/usr/lib/security/$ISA/libpam_ldap.so.1

说明

LDAP 的 PAM 服务模块 **/usr/lib/security/$ISA/libpam_ldap.so.1** 提供了所有四个 PAM 模块的功能，这些模块分别为：身份验证、帐户管理、会话管理和口令管理。

libpam_ldap.so.1 模块是一个共享对象，可以根据需要动态加载以提供所需的功能。其路径在 PAM 配置文件中指定。

LDAP 身份验证模块

LDAP 身份验证组件提供了相应函数，用于验证用户身份 (**pam_sm_authenticate()**) 和设置特定于用户的凭据 (**pam_sm_setcred()**) 。

pam_sm_authenticate() 将用户输入的口令与 LDAP 目录服务器中的口令进行比较。如果口令匹配，用户将通过验证。

可将下列选项传递给 UNIX 服务模块：

调试 — **syslog() LOG_DEBUG** 级别的调试信息。请参阅 *syslog*(3)。

nowarn — 关闭警告消息。

use_first_pass — 将口令数据库中的口令与用户的初始口令（用户验证到堆栈中第一个身份验证模块时输入的口令）进行比较。如果口令不匹配，或者未输入任何口令，则退出且不提示用户输入口令。

仅当在 **pam.conf** 配置文件中将验证服务指定为 **optional** 时才能使用此选项。

try_first_pass — 将口令数据库中的口令与用户的初始口令（用户验证到堆栈中第一个身份验证模块时输入的口令）进行比较。如果口令不匹配，或者未输入任何口令，则提示用户输入口令。

ignore_unknown — 对于 LDAP 储备库中未找到的用户，该标记将强制 **pam_ldap** 的身份验证模块返回 [PAM_IGNORE] 而不是 [PAM_USER_UNKNOWN] 。仅当为本地用户启用了 *pam_hpsec*(5) 中的 **AUTH_MAXTRIES** ，并且在 **pam.conf** 配置文件中的 **pam_unix** 之后配置了 **pam_ldap** 时，才应设置此标记。

当提示输入当前口令时，LDAP 身份验证模块将使用提示符： **Password:** 。

pam_sm_setcred() 函数可设置特定于用户的凭据。对于 LDAP，这是一个 NULL 函数。

LDAP 帐户管理模块

LDAP 帐户管理组件提供了一个可执行帐户管理的函数 (**pam_sm_acct_mgmt()**) 。该函数从身份验证过程中设置的 pam 头中检索数据，该身份验证过程将指示口令是否已在目录服务器中过期。

调试 **syslog() LOG_DEBUG** 级别的调试信息。

nowarn 关闭警告消息。

rcommand 某些 HP-UX 版本要求对 **r**-command 使用此选项，例如 *rlogin*(1)，以便与 PAM 一起使用。

警告：如果启用 **rcommand** 选项，在远程主机上具有活动帐户的用户将可以 **rlogin** 到本地主机或禁用的帐户。

LDAP 会话管理模块

LDAP 会话管理组件提供了相应函数，用于启动 (**pam_sm_open_session()**) 和终止 (**pam_sm_close_session()**) LDAP 会话。对于 LDAP， **pam_open_session()** 是一个 NULL 函数。可将下列选项传入 LDAP 服务模块：

调试 **syslog() LOG_DEBUG** 级别的调试信息。

nowarn 关闭警告消息。

pam_close_session 是一个 NULL 函数。

LDAP 口令管理模块

LDAP 口令管理组件提供了一个函数 (**pam_sm_chauthtok()**)，用于更改 LDAP 目录服务器中的口令。此模块在 **pam.conf** 中必须是 **required** ，而不能是 **optional** 或 **sufficient** 。可将下列选项传入 LDAP 服务模块：

调试 **syslog() LOG_DEBUG** 级别的调试信息。

nowarn 关闭警告消息。

use_first_pass 将口令数据库中的口令与用户的旧口令（在堆栈的第一个口令模块中输入的口令）进行比较。如果口令不匹配，或者未输入任何口令，则退出且不提示用户输入旧口令。它还会尝试使用新口令（在堆栈的第一个口令模块中输入的口令）作为此模块的新口令。如果新口令失败，则退出且不提示用户输入新口令。

try_first_pass 将口令数据库中的口令与用户的旧口令（在堆栈的第一个口令模块中输入的口令）进行比较。如果口令不匹配，或者未输入任何口令，则提示用户输入旧口令。它还会尝试使用新口令（在堆栈的第一个口令模块中输入的口令）作为此模块的新口令。如果新口令失败，则提示用户输入新口令。

如果用户的口令已过期，LDAP 帐户模块将使用 **pam_set_data()** 在身份验证句柄中保存此信息。LDAP 口令模块使用 **pam_get_data()** 从身份验证句柄中检索此信息，以确定是否要强制用户更新其口令。

另请参阅

pam(3)、 pam_authenticate(3)、 pam_setcred(3)、 syslog(3)、 pam.conf(4)、 pam_user.conf(4)、 ldapux(5)。

名称

pam_unix - UNIX 的验证、帐户、会话和口令管理 PAM 模块

概要

/usr/lib/security/$ISA/libpam_unix.so.1

说明

UNIX 的 PAM 服务模块 **/usr/lib/security/$ISA/libpam_unix.so.1** 提供了所有四个 PAM 模块的功能：验证、帐户管理、会话管理和口令管理。

libpam_unix.so.1 模块是一个共享对象，可以根据需要动态加载以提供所需的功能。

有关模块路径的解释，请参考 *pam.conf*(4) 中的相关信息。

Unix 验证模块

UNIX 验证组件提供了相应函数，用来验证用户身份 (**pam_sm_authenticate()**) 和设置特定于用户的凭证 (**pam_sm_setcred()**) 。

pam_sm_authenticate() 将用户输入的口令（或从用户的智能卡中检索出来的口令）与来自 UNIX 口令数据库的口令（包括受信任系统的受保护口令数据库）相比较。如果口令匹配，则用户通过验证。如果用户还具有安全 RPC 凭证，并且安全 RPC 口令与 UNIX 口令相同，则会同时获取该安全 RPC 凭证。

可将下列选项传递给 UNIX 服务模块：

debug **LOG_DEBUG** 级别的 *syslog*(3C) 调试信息。

nowarn 关闭警告消息。

use_first_pass 它比较口令数据库中的口令与用户的初始口令（用户向堆栈中的第一个验证模块进行验证时输入的口令）。如果口令不匹配，或者如果未输入任何口令，则退出并且不提示用户输入口令。仅当在 **pam.conf** 配置文件中将验证服务指定为 *optional* 时才能使用此选项。

try_first_pass 它比较口令数据库中的口令与用户的初始口令（用户向堆栈中的第一个验证模块进行验证时输入的口令）。如果口令不匹配，或者如果未输入任何口令，则提示用户输入口令。

use_psd psd 代表个人安全设备；当前实现中只有一个安全设备：智能卡。它比较口令数据库中的口令与用户智能卡中存储的口令。在此选项下，系统将使用 PAM Framework 提示“Enter PIN:”，而不使用口令提示。此选项仅适用于 **pam.conf** 或 **pam_user.conf** 配置文件中的验证或口令模块类型（auth、口令）服务。

当提示输入当前口令时，UNIX 验证模块将使用提示“Password:”，除非发生以下情况之一：

1. 指定了 **try_first_pass** 选项，并且为堆栈中的第一个模块输入的口令不能通过 UNIX 模块的验证。
2. 未指定 **try_first_pass** 选项，并且 **pam.conf** 文件中早先列出的验证模块已提示用户输入口令。
3. 指定了 **use_psd** 选项。在这种情况下，UNIX 验证模块将使用提示“Enter PIN:”。

在第 1 和第 2 种情况下，UNIX 验证模块将使用提示“System Password:”。

pam_sm_setcred() 函数用来设置特定于用户的凭证。如果用户具有安全 RPC 凭证，但安全 RPC 口令与 UNIX 口令不相同，则会显示一条警告消息。如果用户希望获取安全 RPC 凭证，则需要运行 *keylogin*(1) 。

Unix 帐户管理模块

UNIX 帐户管理组件提供了一个可执行帐户管理的函数 (**pam_sm_acct_mgmt()**) 。该函数从 UNIX 口令数据库中检索用户的口令条目，并验证用户的帐户和口令是否过期。对于受信任系统，此模块还会基于安全配置验证允许的访问时间和访问终端。以下选项可以传递给该 UNIX 服务模块：

debug **LOG_DEBUG** 级别的 *syslog*(3C) 调试信息。

nowarn 关闭警告消息。

Unix 会话管理模块

UNIX 会话管理组件提供了相应函数，用来启动 (**pam_sm_open_session()**) 和终止 (**pam_sm_close_session()**) UNIX 会话。对于 UNIX，在受信任模式下， **pam_open_session()** 将更新受保护口令数据库库中上一次成功或不成功的登录时间。帐户管理模块将读取该信息以显示用户上一次登录的时间。

以下选项可以传递给该 UNIX 服务模块：

debug **LOG_DEBUG** 级别的 *syslog*(3C) 调试信息。

nowarn 关闭警告消息。

pam_close_session 是一个 NULL 函数。

Unix 口令管理模块

UNIX 口令管理组件提供了一个可以更改 UNIX 口令数据库中的口令的函数 (**pam_sm_chauthtok()**) 。此模块在 **pam.conf** 中必须是 **required** ，而不能是 **optional** 或 **sufficient** 。以下选项可以传递给该 UNIX 服务模块：

debug **LOG_DEBUG** 级别的 *syslog*(3C) 调试信息。

nowarn 关闭警告消息。

use_first_pass 它比较口令数据库中的口令与用户的旧口令（为堆栈中第一个口令模块输入的口令）。如果口令不匹配，或者如果未输入任何口令，则退出并且不提示用户输入旧口令。它还会尝试使用新口令（为堆栈中第一个口令模块输入的口令）作为此模块的新口令。如果新口令失败，则退出并且不提示用户输入新口令。

try_first_pass 它比较口令数据库中的口令与用户的旧口令（为堆栈中第一个口令模块输入的口令）。如果口令不匹配，或者如果未输入任何口令，则提示用户输入旧口令。它还会尝试使用新口令（为堆栈中第一个口令模块输入的口令）作为此模块的新口令。如果新口令失败，则提示用户输入新口令。

use_psd 它提示用户输入 PIN（使用 PIN，PAM Framework 可以从智能卡中检索口令）并从智能卡中检索旧口令。然后它将口令数据库中的口令与用户的旧口令相比较。如果口令匹配，则提示用户输入新口令。

如果用户的口令已过期，UNIX 帐户模块将使用 **pam_set_data()** 在验证处理程序中保存此信息。UNIX 口令模块使用 **pam_get_data()** 从验证处理程序中检索此信息，确定是否要强制用户更新其口令。

实际应用信息

在受信任系统上，UNIX 服务模块 **libpam_unix** 中实现的 **pam_sm_*()** 接口不具有线程安全性。除此之外，它们具有线程安全性。当某个线程执行任何这些接口时，可能会出现一个取消点。它们不具有取消安全性和异步取消安全性，也不具有异步信号安全性。

警告

HP-UX 11i v3 是支持受信任系统功能的最后一个版本。

另请参阅

keylogin(1)、 pam(3)、 pam_authenticate(3)、 pam_setcred(3)、 syslog(3C)、 nsswitch.conf(4)、 pam.conf(4)、 pam_user.conf(4)。

名称

pam_updbe - 用户策略定义服务模块

概要

/usr/lib/security/libpam_updbe.1

说明

PAM 的用户策略定义服务模块 **/usr/lib/security/libpam_updbe.1** 读取定义在用户配置文件 **/etc/pam_user.conf** （请参阅 *pam_user.conf*(4) ）中的 选项，并使用 **pam_set_data** （请参阅 *pam_set_data*(3) ）在 *pam* 句柄中存储信息，供后继服务模块使用。当验证用户时或者用户口令被更改时，服务模块执行 **pam_get_data** ，以检索 *pam* 句柄中的相应信息。

并不强制要求使用 **pam_updbe** 。仅当使用每个用户配置时，才需要使用它。但是，要想使 **pam_updbe** 的功能生效，必须在 **/etc/pam.conf** 中将它列为 **pam_hpsec** 之后的第一个服务模块。

与其他服务模块类似， **pam_updbe** 提供所有四个 PAM 模块的接口。这四个模块是验证、帐户管理、会话管理和口令管理。每个模块仅读取为特定的模块类型而定义的选项。

UPDBE 验证模块

UPDBE 验证组件提供多个函数，为模块类型“auth”读取定义在 **pam_user.conf** 中的选项。所使用的模块数据名称是 **PAM_AUTH_USER** 。

Unix 帐户管理模块

UNIX 帐户管理组件提供一个函数，为模块类型“account”读取定义在 **pam_user.conf** 中的选项。所使用的模块数据名称是 **PAM_ACCOUNT_USER** 。

Unix 会话管理模块

UNIX 会话管理组件提供一个函数，为模块类型“session”读取定义在 **pam_user.conf** 中的选项。所使用的模块数据名称是 **PAM_SESSION_USER** 。

Unix 口令管理模块

UNIX 口令管理组件提供一个函数，为模块类型“password”读取定义在 **pam_user.conf** 中的选项。所使用的模块数据名称是 **PAM_PASSWORD_USER** 。

另请参阅

pam(3)、 pam_set_data(3)、 pam.conf(4)、 pam_user.conf(4)、 pam_hpsec(5)。

名称

partition - 显示分区命令行接口的有关信息

概要

partition

说明

本联机帮助页提供一个简短的列表说明，介绍用于管理分区系统的命令。

命令	说明
cplxmodify	修改现有组合系统。
parcreate	创建一个新分区。
parmodify	修改现有分区。
parstatus	显示可分区系统的分区和可用资源信息。
parremove	删除现有分区。
parunlock	解锁稳定组合系统配置数据或分区配置数据。
fruled	打开（或关闭）单元、机柜和 I/O 机箱的警示指示灯。
frupower	打开（或关闭）单元和 I/O 机箱的电源。

作者

partition 由 HP 开发。

另请参阅

fruled(1)、 parstatus(1)、 cplxmodify(1M)、 frupower(1M)、 parcreate(1M)、 parmgr(1M)、 parmodify(1M)、 parremove(1M)、 parunlock(1M)。

http://docs.hp.com 上的《nPartition Administrator’s Guide》。

名称

pci_eh_enable - 启用（或禁用）PCI 错误恢复

值

保证安全

1

缺省值

1 （启用 PCI 错误恢复）。

允许值

0 至 **1** 之间的整数值。

建议值

1 （启用 PCI 错误恢复）。

说明

HP-UX 支持 PCI 错误处理（或恢复），以确保故障卡导致的 PCI 错误不会使高端系统上的硬盘分区崩溃。该功能有助于恢复最常见的奇偶错误。该可调参数为静态参数，因为很难在运行时修改硬件功能。

pci_eh_enable 的属性是公用 (public) 的，因此某些 Service Guard 用户能够关闭该功能。

更改此可调参数的人员

建议由 Service Guard 用户更改。

更改限制

对此可调参数的更改将在下次重新引导时生效。

应该何时启用此可调参数选项

要启用 PCI 错误恢复功能时，应启用此可调参数

应该何时禁用此可调参数选项

要禁用 PCI 错误恢复功能时

禁用此可调参数的负面影响

将禁用 PCI 错误恢复功能

警告

如果平台不具备此功能，那么，即使可调参数的状态为 **ON** ，也将无法使用 PCI 错误处理（或恢复）功能。

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

pci_eh_enable 由 HP 开发。

名称

pci_error_tolerance_time - I/O 插槽上两个 PCI 错误之间的时间间隔（以分钟为单位），将导致自动 PCI 错误恢复

值

缺省值

1440 （24 小时）

允许值

0 （禁用）或非零（启用）

建议值

1440

说明

pci_error_tolerance_time 是与以分钟为单位的时间间隔对应的动态可调参数。它决定在 I/0 插槽上发生的两个 PCI 错误之间是否要启动自动 PCI 错误恢复。如果在指定的时间间隔内，在插槽上发生两个 PCI 错误，则插槽中的 PCI 卡将挂起，并且必须执行手动 PCI 错误恢复操作才能使卡正常工作。有关手动 PCI 错误恢复操作的详细信息，请参考 *olrad*(1M) 联机帮助页中的"PCI Error Handling"一节。

更改此可调参数的人员

系统管理员可以根据系统中显示的 PCI 错误频率更改 **pci_error_tolerance_time** 的值。

更改限制

无。此可调参数是动态可调参数。对此可调参数的更改立即生效。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

pci_error_tolerance_time 由 HP 开发。

另请参阅

olrad(1M)。

"PCI Error Handling"可从 **http://docs.hp.com** 上的"High Availability"下获得。

已过时

名称

pfdat_hash_locks - 已过时的内核可调参数

说明

pfdat_hash_locks 可调参数已过时并被删除。 HP-UX 将根据系统配置自动计算该值。

多个进程同时访问或修改包含了正在运行进程信息或内存使用情况信息的全局内核结构，这种情况比较常见。要想防止发生争用，必须使用 spinlock （同步所使用的内核数据）对这些结构进行保护， spinlock 仅允许 spinlock "持有者"继续访问，而其他对结构的访问尝试都必须等待。

当需要保护此类数据结构的每个实例，而且有多个实例时，将使用散列的 spinlock。所有实例仅使用一个 spinlock 将导致过多的争用，但是每个结构使用一个 spinlock 将导致内存浪费，而且在任何给定时刻，大多数锁都没有被使用。

通过分配一个散列锁池，散列函数为每组结构选择一个锁，从而减少了争用内存的情况，并节省了内存。系统计算的 **pfdat_hash_locks** 值设定了这种针对 pfdat 数据结构 spinlock 的池的大小。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

pfdat_hash_locks 由 HP 开发。

名称

physical_io_buffers - 物理 I/O 操作的缓冲区总计

说明

此可调参数在 HP-UX 11i v1.6 和 2.0 中使用，可调整内核中物理 I/O 操作所使用的缓冲区组成的共享池的大小。

在 HP-UX 11i v3.0 及更高版本中，内核将自动管理该池的大小。不再需要此可调参数，从 HP-UX 11i v3.0 起此可调参数已过时。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

physical_io_buffers 由 HP 开发。

名称

portal - 应用程序的“未来之窗”

概要

#include <sys/portal.h>

说明

此头文件是应用程序的“未来之窗”。它可以帮助用户:

- 编写跨 32 位系统和 64 位系统的可移植代码,
- 避免整数型大小的未记录假定,
- 编写需要显式声明整数型的大小的可移植代码,
- 编写可以移植到整数型大小不同的平台的代码,和
- 共享跨 32 位系统和 64 位系统的可移植常用宏。

除了定义在此文件中的宏之外,它还包括头文件 **limits.h** (请参阅 *limits*(5))和 **inttypes.h** (请参阅 *inttypes*(5))。

以下宏定义在 **sys/portal.h** 中:

SET_MASK_BIT(*bit_num*, *type*)
这个宏可用于创建具有一个位集的掩码, *bit_num* 是要设置的位的位置, *type* 是掩码的数据类型。如果发生上溢和下溢,将返回 -1。

SET_MASK_BIT_LOOP(*mask*, *bit_num*, *type***)**
这个宏可用于设置掩码中的一个位。 *mask* 是掩码的当前值, *bit_num* 是要设置位的位置, *type* 是掩码的数据类型。

SIGN_BIT(*type***)**
这个宏可用于返回特定数据类型的符号位的位位置, *type* 是数据类型,需要返回它的符号位的位置。

SIGN_BIT_MASK(*type***)**
这个宏可用于返回特定数据类型的符号位掩码, *type* 是数据类型,需要返回它的符号位掩码。

SIGN_EXTEND(*value*, *old_type*, *new_type***)**
这个宏可用于进行从一个数据类型到另一个数据类型的符号扩展, *value* 是要进行符号扩展的当前值, *old_type* 是 *value* 的当前数据类型, *new_type* 是 *value* 的新数据类型。

TEST_ENDIAN(*endian***)**
这个宏可用于检查代码编译为 big endian 还是 little endian。 *endian* 是一个整数,它包含将要返回的结果。

以下宏可用于各个数据类型的值的打印格式和扫描格式;数据类型的大小可能根据编译标志 **_FILE_OFFSET_BITS** 而改变。此类数据类型的示例包括 *off_t* 和 *fpos_t* 。

PRIdF64 **d** 输出 32 位或 64 位大小值的格式化选项。

PRIoF64 **o** 输出 32 位或 64 位大小值的格式化选项。
PRIxF64 **x** 输出 32 位或 64 位大小值的格式化选项。
PRIuF64 **u** 输出 32 位或 64 位大小值的格式化选项。

SCNdF64 **d** 扫描 32 位或 64 位大小值的格式化选项。
SCNoF64 **o** 扫描 32 位或 64 位大小值的格式化选项。
SCNxF64 **x** 扫描 32 位或 64 位大小值的格式化选项。
SCNuF64 **u** 扫描 32 位或 64 位大小值的格式化选项。

举例

以下示例中的 **SET_MASK_BIT** 宏将打开 64 位整数中的高位。

```
SET_MASK_BIT(SIGN_BIT(int64_t), int64_t)
```

以下示例中的 **SET_MASK_BIT** 宏将用于打开 32 位整数中除符号位以外的所有位。

```
~SET_MASK_BIT(SIGN_BIT(int32_t), int32_t)
```

以下示例中的 **SET_MASK_BIT_LOOP** 宏将打开最大整数的重要性最小的三个位。

```
int i;
intmax_t mask = 0;

for (i = 0; i < 3; i++) {
   SET_MASK_BIT_LOOP(mask, i, intmax_t);
}
```

以下示例中的 **SIGN_BIT** 宏用于返回 32 位整数中符号位的位置。

```
SIGN_BIT(int32_t)
```

以下示例中的 **SIGN_BIT_MASK** 宏将返回 32 位整数的符号位掩码。

```
SIGN_BIT_MASK(int32_t)
```

以下示例中的 **SIGN_EXTEND** 宏将存储为字符数据类型的 8 位整数转换为 64 位整数，并正确地扩展符号。

```
char c;
int64_t i;

i = SIGN_EXTEND(c, char, int64_t);
```

如果编译为 big endian，以下示例中的 **TEST_ENDIAN** 宏将在 *endian* 中存储 1；否则，它将在 *endian* 中存储 0。

```
int endian;
```

```
TEST_ENDIAN(endian);

if (endian == 0)
   printf("This a little endian system\n");

if (endian == 1)
   printf("This a big endian system\n");
```

作者

portal.h 由 HP 开发。

文件

/usr/include/sys/portal.h

另请参阅

inttypes(5)、 limits(5)、 printf(3S)、 scanf(3S)。

名称

privgrp - HP-UX 组权限

说明

HP-UX 允许将类似超级用户的有限权限转分给所有用户或特定组的成员。不推荐使用此功能，并且只有现有应用程序才能使用此功能。新应用程序应当使用在 *privileges*(5) 中说明的较新的精细划分的权限功能。

<sys/privgrp.h> 头文件定义了以下符号权限名称： **PRIV_CHOWN** 、 **PRIV_FSSTHREAD** 、 **PRIV_LOCKRDONLY** 、 **PRIV_MLOCK** 、 **PRIV_MPCTL** 、 **PRIV_PSET** 、 **PRIV_RTPRIO** 、 **PRIV_RTSCHED** 、 **PRIV_SERIALIZE** 、 **PRIV_SETRUGID** 和 **PRIV_SPUCTL** 。

除一个组权限之外，支持所有组权限作为精细划分的权限，如 权限(5) 所述。不作为精细划分的权限支持的一个组权限是：

PRIV_SETRUGID

允许使用 **setuid()** 和 **setgid()** 系统调用分别更改一个进程的实际用户 ID 和实际组 ID （请参阅 *setuid*(2) ）。不推荐采用 **setuid()** 的此行为，仅传统应用程序才可采用。较新的应用程序应当分别使用 **setresuid(geteuid(), -1, -1)** 和 **setresgid(getegid(), -1, -1)** 以达到相同的效果（不需要专用权限）。

<sys/privgrp.h> 头文件定义了两个附加符号常量：

PRIV_MAXGRPS

定义具有专用权限的最大数量的组。在此最大数量的组中，有一个组是为全局权限而保留的（已授予给所有进程），而剩下的组可分配给实际的组 ID。

PRIV_MASKSIZ

定义多字掩码的大小，这些掩码在定义与组 ID 相关联的权限时使用。

setprivgrp 和 **getprivgrp** 命令，以及 **setprivgrp()** 和 **getprivgrp()** 系统调用可用于定义和查询权限组关联。

组权限在引导时从 **/etc/privgroup** 的内容自动初始化（请参阅 *privgrp*(4) ）。

警告

不推荐使用此机制，并且只有传统应用程序才可使用此机制。有关精细划分的权限的说明，请参阅 *privileges*(5)。

另请参阅

getprivgrp(1) 、 setprivgrp(1M) 、 chown(2) 、 getprivgrp(2) 、 lockf(2) 、 mpctl(2) 、 plock(2) 、 pset_create(2) 、 rtprio(2)、 rtsched(2)、 serialize(2)、 setgid(2)、 setuid(2)、 shmctl(2)、 privgrp(4)、 privileges(5)。

名称

privileges - HP-UX 权限说明

说明

UNIX® 操作系统传统上惯用“全部或零”权限模型，超级用户（拥有有效的 **UID 0**，例如，名为 **root** 的用户）权限几乎不受任何限制，而其他用户只有很少或没有特殊权限。

系统管理员通常需要授权给其他用户有限的权限。 HP-UX 提供了多种授权方法。由于这些机制允许除了超级用户以外的其他用户执行某些需要权限的操作，因此，HP-UX 文档在描述允许执行某操作的用户时，通常使用诸如“授权用户”或“拥有相应权限的用户”等术语，而不使用“超级用户”。

如果没有具体说明执行某操作所需要的权限（通常位于该操作的联机帮助页中），通常可以假定超级用户享有适当的权限。

传统的授权方法

HP-UX 使用多种方法来授予有限的权限，其中包括受限的 **sam**、 *privgrp*(5)中介绍的权限组、 *shutdown*(1M)中介绍的 **shutdown.allow** 文件以及 *crontab*(1)中介绍的 **cron.allow** 文件。

精细划分的权限

HP-UX 精细划分的权限模型将超级用户的权限划分成了一组权限。其中每个权限均为拥有该权限的进程授予了执行特定的一组由内核提供的受限服务的权限。通过“权限调整”，进程可以从内部管理这些权限。“权限调整”指的是，仅在需要某权限时启用或“提升”该权限，然后再禁用或“降低”该权限。进程提升的权限用于确定该进程可以调用的敏感系统调用服务。

传统权限

“传统权限”是那些最初在 *privgrp*(5) 中定义的权限。该权限集中除 **PRIV_SETRUGID** 之外的所有权限均已合并到精细划分的权限，如：

PRIV_CHOWN PRIV_FSSTHREAD PRIV_LOCKRDONLY PRIV_MLOCK
PRIV_MPCTL PRIV_PSET PRIV_RTPRIO PRIV_RTSCHED
PRIV_SERIALIZE PRIV_SPUCTL

基本权限

缺省情况下，所有进程都将授予 “基本权限”。基本权限包含下列权限：

PRIV_EXEC PRIV_FORK PRIV_LINKANY PRIV_SESSION

根替换权限

“根替换权限”可以提供与有效用户 ID 为零的进程相关联的权限。根替换权限包括：

PRIV_ACCOUNTING PRIV_AUDCONTROL PRIV_CHOWN PRIV_CHROOT
PRIV_CHSUBJIDENT PRIV_DACREAD PRIV_DACWRITE PRIV_DEVOPS
PRIV_DLKM PRIV_FSINTEGRITY PRIV_FSS PRIV_FSSTHREAD
PRIV_LIMIT PRIV_LOCKRDONLY PRIV_MKNOD PRIV_MLOCK
PRIV_MOUNT PRIV_MPCTL PRIV_NETADMIN PRIV_NETPRIVPORT
PRIV_NETPROMISCUOUS PRIV_NETRAWACCESS PRIV_OBJSUID PRIV_OWNER

PRIV_PSET PRIV_REBOOT PRIV_RTPRIO PRIV_RTSCHED
PRIV_RTPSET PRIV_SELFAUDIT PRIV_SERIALIZE PRIV_SPUCTL
PRIV_SYSATTR PRIV_SYSNFS

缺省情况下，会将这些权限授予任何有效用户 ID 为零的进程。

策略覆盖权限

“策略覆盖权限”可覆盖隔离专区规则。有四种策略覆盖权限：

PRIV_CHANGECMPT PRIV_CMPTREAD PRIV_CMPTWRITE PRIV_COMMALLOWED.

缺省情况下，不会将这些权限授予有效用户 ID 为零的进程。这些权限仅适用于隔离专区功能（请参阅 *compartments*(5) 和 *cmpt_tune*(1M) 以确定是否已启用该功能）。这些权限是复合权限 **POLICY** 中的权限集合的组成部分。

策略配置权限

“策略配置权限”用于控制如何配置权限。这样的权限有两个：**PRIV_CHANGEFILEXSEC** 和 **PRIV_RULESCONFIG**。缺省情况下，不会将这两个权限授予有效用户 ID 为零的进程。这两个权限是复合权限 **POLICY** 中的一组权限的组成部分。

进程属性权限

“进程属性权限”仅仅在操作方式上与其他权限相同。**PRIV_TRIALMODE** 是此组权限的唯一成员。缺省情况下，不会将此权限授予有效用户 ID 为零的进程。

复合权限

“复合权限”是一种用来指定一组预定义的简单权限的便捷方式。在将来的版本中可能会重新定义这些复合权限，以便创建新的权限。复合权限定义如下：

BASIC 指基本权限。

BASICROOT 指基本权限和根替换权限的组合。

POLICY 指策略覆盖权限和策略配置权限。

权限说明

下面的列表指定了权限名称及其主要用途。

PRIV_ACCOUNTING (ACCOUNTING)
允许进程控制进程记账系统（请参阅 *acct*(2)）。

PRIV_AUDCONTROL (AUDCONTROL)
允许进程启动、修改和停止审核系统。

PRIV_CHANGECMPT (CHANGECMPT)
授予进程修改其隔离专区的能力（请参阅 *compartments*(5) 和 *cmpt_tune*(1M) 以确定是否已启用此扩展功能）。

PRIV_CHANGEFILEXSEC (CHANGEFILEXSEC)

允许进程向二进制代码授予权限。

PRIV_CHOWN (CHOWN)

允许访问 **chown()** 系统调用（请参阅 *chown*(2) ）。

PRIV_CHROOT (CHROOT)

允许进程更改其根目录。

PRIV_CHSUBJIDENT (CHSUBJIDENT)

允许进程更改其 UID、GID 和组列表。还允许进程更改 **chown** 文件并保留文件上设置的 suid 或 sgid 位（如果存在）。

PRIV_CMPTREAD (CMPTREAD)

允许进程打开文件或目录以执行读取、执行（为文件时）或搜索（为目录时）操作，而绕过可能会阻止执行该操作的隔离专区规则（请参阅 *compartments*(5) 和 *cmpt_tune*(1M) 以确定是否已启用此扩展功能）。

PRIV_CMPTWRITE (CMPTWRITE)

允许进程向文件或目录执行写入操作，而绕过可能会阻止执行该操作的隔离专区规则（请参阅 *compartments*(5) 和 *cmpt_tune*(1M) 以确定是否已启用此扩展功能）。

PRIV_COMMALLOWED (COMMALLOWED)

允许进程覆盖 IPC 和网络子系统中的隔离专区规则（请参阅 *compartments*(5) 和 *cmpt_tune*(1M) 以确定是否已启用此扩展功能）。

PRIV_DACREAD (DACREAD)

允许进程覆盖所有任意的读取、执行和搜索访问限制。有关详细信息，请参阅“任意的限制”。

PRIV_DACWRITE (DACWRITE)

允许进程覆盖所有任意的写入访问限制。有关详细信息，请参阅“任意的限制”。

PRIV_DEVOPS (DEVOPS)

允许进程执行设备特定的管理操作，例如，磁带或磁盘格式设置。

PRIV_DLKM (DLKM)

允许进程加载内核模块（请参阅 *modload*(2) ）、获取有关已加载的内核模块的信息（请参阅 *modstat*(2) ）以及更改可动态加载的内核模块的全局搜索路径（请参阅 *modpath*(2) ）。

PRIV_EXEC (EXEC)

允许进程调用 **exec()** （请参阅 *exec*(2) ）系列调用。

PRIV_FORK (FORK)

允许进程创建其他的进程（使用 **fork()** 和 **vfork()** ）。

PRIV_FSINTEGRITY (FSINTEGRITY)
允许进程执行各项磁盘操作，如删除或修改磁盘分区的尺寸或边界等，或在系统中导入和导出 LVM 卷组。

PRIV_FSSTHREAD (FSSTHREAD)
保留。

PRIV_FSS (FSS)
保留。

PRIV_LIMIT (LIMIT)
允许进程设置最大限制值之外的资源和优先级限制（请参阅 *setrlimit*(2) 或 *nice*(2)）。

PRIV_LINKANY (LINKANY)
保留。

PRIV_LOCKRDONLY (LOCKRDONLY)
允许使用 **lockf()** 系统调用对打开的只读文件设置锁定（请参阅 *lockf*(2)）。

PRIV_MKNOD (MKNOD)
允许进程使用 **mknod()** 系统调用创建字符或块设备专用文件（请参阅 *mknod*(2)）。

PRIV_MLOCK (MLOCK)
允许访问 **plock()** 系统调用（请参阅 *plock*(2)）。

PRIV_MOUNT (MOUNT)
允许进程使用 **mount()** 和 **umount()** 系统调用挂接和卸除文件系统。请参阅 *mount*(2) 和 *umount*(2)。

PRIV_MPCTL (MPCTL)
允许使用 **mpctl()** 系统调用来更改进程的处理器绑定、位置域绑定或启动策略（请参阅 *mpctl*(2)）。

PRIV_NETADMIN (NETADMIN)
允许进程执行网络管理操作，其中包括配置网络路由表和查询接口信息。

PRIV_NETPRIVPORT (NETPRIVPORT)
允许将进程绑定到权限端口。缺省情况下，端口号 **0-1023** 为权限端口。

PRIV_NETPROMISCUOUS (NETPROMISCUOUS)
允许进程配置接口以便用混合模式进行监听。

PRIV_NETRAWACCESS (NETRAWACCESS)
允许进程访问原始的 Internet 网络协议。

PRIV_OBJSUID (OBJSUID)

如果进程拥有 **OWNER** 权限，则可允许进程在任何文件上设置 suid 和 sgid 位。此外，如果该进程有权限修改文件所有权，则进程不需要清除 suid 或 sgid 位即可更改文件的所有权。

PRIV_OWNER (OWNER)

允许进程覆盖与文件或资源的所有者相匹配的 UID 相关的所有限制。有关详细信息，请参阅"任意的限制"。

PRIV_PSET (PSET)

允许更改系统处理器集配置（请参阅 *pset_create*(2)）。

PRIV_REBOOT (REBOOT)

允许进程执行重新引导操作。

PRIV_RTPRIO (RTPRIO)

允许访问 **rtprio()** 系统调用（请参阅 *rtprio*(2)）。

PRIV_RTPSET (RTPSET)

允许进程控制 RTE 处理器集（请参阅 *__pset_rtctl*(2)）。

PRIV_RTSCHED (RTSCHED)

允许访问 **sched_setparam()** 和 **sched_setscheduler()** 来设置 POSIX.4 实时优先级（请参阅 *rtsched*(2)）。

PRIV_RULESCONFIG (RULESCONFIG)

允许进程在系统上添加和修改隔离专区规则。（请参阅 *compartments*(5) 和 *cmpt_tune*(1M) 以确定是否已启用此扩展功能）。

PRIV_SELFAUDIT (SELFAUDIT)

允许进程使用 **audwrite()** 系统调用生成自身的审核记录（请参阅 *audwrite*(2)）。

PRIV_SERIALIZE (SERIALIZE)

允许使用 **serialize()** 强制目标进程与该系统调用所标记的其他进程按串行方式运行（请参阅 *serialize*(2)）。

PRIV_SESSION (SESSION)

允许创建新的会话（请参阅 *setsid*(2)）和 *setpgrp(2)*）。

PRIV_SPUCTL

允许在 Instant Capacity 产品中执行一些管理操作来停用和重新启用处理器。有关详细信息，请参阅 Instant Capacity 文档。

PRIV_SYSATTR (SYSATTR)

允许进程管理系统属性，包括可调参数的设置以及修改主机名、域名和用户配额。

PRIV_SYSNFS (SYSNFS)

允许进程执行 NFS 操作，如导出文件系统、 **getfh()** 系统调用（请参阅 *getfh*(2)）、NFS 文件锁定、撤销 NFS 验证、以及创建 NFS 内核守护程序线程。

PRIV_TRIALMODE (TRIALMODE)

允许进程将试验模式信息记录到 **syslog** 文件。请参阅下面的“试验模式”。

权限编程

在使用权限编程时，与每个权限相关联的名称与此处提供的带有前缀字符串 **PRIV_** 的名称相同（即，在源代码中使用符号常量 **PRIV_ACCOUNTING** ）。在与权限相关联的命令中，尽管绝大多数命令也可以识别带有 **PRIV_** 前缀的名称，但所使用的名称没有该前缀。

复合权限 **BASIC**、 **BASICROOT** 和 **POLICY** 旨在帮助用户更轻松地开发那些即使基础权限发生改变也能保持其功能的应用程序。对于即使一组基础的权限发生改变时也要求保持兼容性的应用程序，应确保不要意外地删除自被开发以来所添加的任何新权限。例如，要做到这一点，可以使用 **priv_remove()** 删除来自有效权限集的特定权限（请参阅 *priv_remove*(3)），或者确保将复合权限用作 **priv_set_effective()** 的参数（请参阅 *priv_set_effective*(3)）。

将权限和二进制代码相关联

依赖于权限的使用的应用程序必须使用 **setfilexsec** 命令进行注册（请参阅 *setfilexsec*(1M)）。有关授予权限的另一种方法的详细信息，请参阅 *privrun*(1M)。

根据应用程序执行的受限任务的类型，应用程序可以在执行任务之前提升所需要的相应权限，然后在完成任务后降低该权限。这种做法称为 权限调整。建议进程在任何给定的时间运行时尽可能的使用最小权限设置。

将权限与进程相关联

每个进程都有三个与之相关联的权限集。这些权限集是:

允许的权限集

进程可以启用的最大权限集。进程可以从该权限集中删除任何权限，但不能向该权限集中添加权限。可以将该权限集的权限添加到进程的有效权限集。该权限集通常也称为可能的权限集。

有效权限集

进程当前处于活动状态的权限集。进程可以在任何给定时间修改此权限集，使其仅保留必要的权限。可以删除此权限集中的任何权限，但只可以添加进程允许的权限集中的权限。进程的有效权限集始终是其允许的权限集的子集。

保留的权限集

进程调用 **execve()** 时保留的权限集（请参阅 *execve*(2)）。进程可以删除该权限集中的任何权限，但不能向该权限集中添加任何权限。进程的保留的权限集始终是其允许的权限集的子集。

可以使用函数 **priv_add_effective()**、 **priv_remove()** 和 **priv_get()** 中指定的库调用来管理这些权限集（请参阅 *priv_add_effective*(3)、 *priv_remove*(3) 和 *priv_get*(3)）。

任意的限制

任意的限制是由传统的文件模式访问权限施加的限制。因此，即便文件模式权限禁止，权限 **PRIV_DACREAD** 和 **PRIV_DACWRITE** 也允许执行读取、搜索、执行和写入操作。 **PRIV_OWNER** 权限允许那些不是文件或目录的所有者的进程删除其父目录设置了粘着位的文件或目录。 **PRIV_OWNER** 权限还允许那些不是 System V IPC 消息队列、信号量集、或共享内存段的所有者的进程，删除、修改这些对象的所有者，或者修改它们的权限位。

试验模式

这是系统提供的一项功能，用于帮助报告进程在其生命周期内已使用的权限列表。开发人员可以使用此功能验证应用程序运行所需要的权限。每当拥有此权限的进程尝试使用任何权限时（通过执行使用该权限的系统调用），会将一个条目记录到 **syslog** ，将这些条目汇总到一起便构成一个已用权限的列表。

兼容性

有效用户 ID 为零的进程，在缺省情况下，将被视为拥有根替换权限的进程。设置隔离专区功能可能会进一步限制对有效用户 ID 的解释，使得进程好像只拥有根替换权限的一个指定子集。有关详细信息，请参阅 *compartments*(4) 中的“进程限制规则”说明。

更正式的说法是，当且仅当满足下列一个或多个条件，进程便将拥有一项权限：

- 权限出现在其有效权限集中，或者
- 权限为根替换权限，进程的有效用户 ID 为零，且未启用设置隔离专区功能，或者
- 权限为根替换权限，进程的有效用户 ID 为零，启用设置隔离专区功能，并在进程的隔离专区中可以使用该权限。

系统权限要求

本节提供的表列出了相应的联机帮助页在指定“相应权限”来执行某些操作或按某些条件运行时可能需要的权限。对于每个系统调用，此表还列出了可能会影响系统调用的行为的权限。

本节的各小节还涉及其他的函数和关注的领域。这些表列出了单个联机帮助页在指定“相应权限”来执行某些操作或按某些条件运行时可能需要的权限。

多个系统调用可供有权限和没有权限的应用程序访问。例如， **kill()** 系统调用（请参阅 *kill*(2) ），当由没有 **PRIV_OWNER** 权限的进程所使用时，仅会向其 UID 与发送进程自身的 UID 相匹配的进程发送信号。

一些通用指导适用于与硬件相关的系统调用。

- 除了具体使用的系统调用所需要的任何权限之外，许多硬件设备还需要 **PRIV_DEVOPS** 权限。
- 对于网络和流而言，除其他权限之外，根据用户试图执行的操作，它们可能还需要 **PRIV_NETADMIN**、 **PRIV_NETRAWACCESS** 和/或 **PRIV_NETPROMISCUOUS** 权限。例如， **exportfs** 命令需要 **PRIV_SYSNFS** 权限（请参阅 *exportfs*(1M) ）。 **fdetach()** 和 **fattach()** 库调用（可能除其他权限之外还）需要 **PRIV_MOUNT** 权限。（请参阅 *fdetach*(3) 和 *fattach*(3C) ）。

用于 **pstat** 系统调用的权限

在对调用进程的隔离专区之外的进程上进行操作时， **pstat()** 系统调用通常需要 **PRIV_COMMALLOWED** 权限（请参阅 *pstat*(2) ）。但是，由于该系统调用应用的区域较多，该调用的一些函数可能需要其他权限。它们需要

的函数和权限如下表所示:

pstat_getcommandline()	**PRIV_COMMALLOWED**
pstat_getfile()/pstat_getfile2()	**PRIV_COMMALLOWED**
pstat_getfiledetails()	**PRIV_COMMALLOWED、PRIV_OWNER**
pstat_getlwp()	**PRIV_COMMALLOWED**
pstat_getmsg()	**PRIV_COMMALLOWED**
pstat_getpmq()	**PRIV_COMMALLOWED**
pstat_getproc()	**PRIV_COMMALLOWED**
pstat_getpsem()	**PRIV_COMMALLOWED**
pstat_getsem()	**PRIV_COMMALLOWED**
pstat_pathname()	**PRIV_COMMALLOWED、PRIV_OWNER**
pstat_proc_locality()	**PRIV_COMMALLOWED**
pstat_proc_vm()	**PRIV_COMMALLOWED**
pstat_procwindow()	**PRIV_COMMALLOWED**
pstat_shminfo()	**PRIV_COMMALLOWED**
pstat_socket()	**PRIV_COMMALLOWED、PRIV_OWNER**
pstat_stream()	**PRIV_COMMALLOWED、PRIV_OWNER**

用于 **Security Containment** 的权限

一些与 Security Containment 相关的命令会使用一些其他环境不使用的权限:

setfilexsec	**PRIV_CHANGEFILEXSEC、PRIV_CMPTREAD、PRIV_CMPTWRITE、PRIV_DACREAD、PRIV_DACWRITE**
setrules	**PRIV_RULESCONFIG**

此外，一些与 Security Containment 相关的库调用还会使用特定的安全权限:

cmpt_change()	**PRIV_CHANGECMPT**
cmpt_get()	**PRIV_COMMALLOWED**
cmpt_get_addrcid()	**PRIV_RULESCONFIG**
cmpt_get_ifcid()	**PRIV_RULESCONFIG**
priv_get()	**PRIV_COMMALLOWED**

privset_get()	**PRIV_COMMALLOWED**

用于系统调用的权限

下表列出了系统调用和它们可能需要的权限。其中一些权限取决于系统调用所作用的系统对象（例如，另一个隔离专区中的文件）、系统的状态（例如，如果打开的文件的数量已达到最大值）或者其他条件。

__pset_rtctl()	**PRIV_PSET**、**PRIV_RTPSET**
accept()	**PRIV_LIMIT**
access()	**PRIV_CMPTREAD** 、 **PRIV_CMPTWRITE** 、 **PRIV_DACREAD** 、 **PRIV_DACWRITE**
acct()	**PRIV_ACCOUNTING**
acl()	**PRIV_CMPTREAD** 、 **PRIV_CMPTWRITE** 、 **PRIV_DACREAD** 、 **PRIV_DACWRITE**、**PRIV_OWNER**
adjtime()	**PRIV_SYSATTR**
audctl()	**PRIV_AUDCONTROL**
audswitch()	**PRIV_SELFAUDIT**
audtag()	**PRIV_SELFAUDIT**
audwrite()	**PRIV_SELFAUDIT**
bind()	**PRIV_NETPRIVPORT**
chdir()	**PRIV_CMPTREAD** 、 **PRIV_CMPTWRITE** 、 **PRIV_DACREAD** 、 **PRIV_DACWRITE**
chmod()	**PRIV_CMPTREAD**、**PRIV_DACREAD**、**PRIV_OWNER**
chown()	**PRIV_CHOWN** 、 **PRIV_CMPTREAD** 、 **PRIV_DACREAD** 、 **PRIV_OWNER**
chroot()	**PRIV_CHROOT**、**PRIV_CMPTREAD**、**PRIV_DACREAD**
clock_settime()	**PRIV_SYSATTR**
connect()	**PRIV_COMMALLOWED**
crashconf()	**PRIV_DEVOPS**
creat()	**PRIV_CMPTREAD** 、 **PRIV_CMPTWRITE** 、 **PRIV_DACREAD** 、 **PRIV_DACWRITE**、**PRIV_LIMIT**、**PRIV_OBJSUID**、**PRIV_OWNER**
dup()	**PRIV_LIMIT**

dup2()	**PRIV_LIMIT**
exec()	**PRIV_CMPTREAD**、**PRIV_DACREAD**、**PRIV_EXEC**
execve()	**PRIV_CMPTREAD**、**PRIV_DACREAD**
fchmod()	**PRIV_OBJSUID**、**PRIV_OWNER**
fchown()	**PRIV_CHOWN**、**PRIV_OWNER**
fork()	**PRIV_FORK**、**PRIV_LIMIT**
fpathconf()	**PRIV_CMPTREAD**、**PRIV_DACREAD**
fsetacl()	**PRIV_OWNER**
ftruncate()	**PRIV_CMPTREAD** 、 **PRIV_CMPTWRITE** 、 **PRIV_DACREAD** 、 **PRIV_DACWRITE**、**PRIV_OBJSUID**、**PRIV_OWNER**
getaccess()	**PRIV_CMPTREAD** 、 **PRIV_CMPTWRITE** 、 **PRIV_DACREAD** 、 **PRIV_DACWRITE**
getacl()	**PRIV_CMPTREAD**、**PRIV_DACREAD**
getaudid()	**PRIV_SELFAUDIT**
getaudproc()	**PRIV_SELFAUDIT**
getevent()	**PRIV_AUDCONTROL**
getfh()	**PRIV_SYSNFS**
getpgrp2()	**PRIV_COMMALLOWED**
getpriority()	**PRIV_COMMALLOWED**
getprivgrp()	**PRIV_SYSATTR**
getsid()	**PRIV_COMMALLOWED**
ioctl()	**PRIV_FSINTEGRITY** 、 **PRIV_SYSATTR** 、 **PRIV_DEVOPS** 、 **PRIV_NETADMIN** 、 **PRIV_NETPROMISCUOUS** 、 **PRIV_NETRAWACCESS** 等。一般情况下， **ioctl** 所需要的权限取决于 **ioctl** 的驱动程序和类型。
kill()	**PRIV_COMMALLOWED**、**PRIV_OWNER**、**PRIV_REBOOT**
lchown()	**PRIV_CMPTREAD**、**PRIV_DACREAD**、**PRIV_OWNER**
link()	**PRIV_CMPTREAD** 、 **PRIV_CMPTWRITE** 、 **PRIV_DACREAD** 、 **PRIV_DACWRITE**、**PRIV_FSINTEGRITY**

lockf()	**PRIV_LOCKRDONLY**
lstat()	**PRIV_CMPTREAD、PRIV_DACREAD**
mem_res_grp()	**PRIV_SYSATTR**
mkdir()	**PRIV_CMPTREAD 、 PRIV_CMPTWRITE 、 PRIV_DACREAD 、 PRIV_DACWRITE、PRIV_LIMIT**
mknod()	**PRIV_CMPTREAD 、 PRIV_CMPTWRITE 、 PRIV_DACREAD 、 PRIV_DACWRITE、PRIV_LIMIT、 PRIV_MKNOD**
mlock()	**PRIV_MLOCK**
mlockall()	**PRIV_MLOCK**
mmap()	**PRIV_DEVOPS**
modload()	**PRIV_CMPTREAD、PRIV_DACREAD、PRIV_DLKM**
modpath()	**PRIV_DLKM**
modstat()	**PRIV_DLKM**
moduload()	**PRIV_DLKM**
mount()	**PRIV_CMPTREAD 、 PRIV_DACREAD 、 PRIV_MOUNT 、 PRIV_OWNER**
mpctl()	**PRIV_COMMALLOWED、PRIV_MPCTL**
mq_open()	**PRIV_COMMALLOWED、PRIV_DACREAD、PRIV_DACWRITE**
mq_unlink()	**PRIV_COMMALLOWED、PRIV_DACREAD、PRIV_DACWRITE**
msgctl()	**PRIV_COMMALLOWED 、 PRIV_DACREAD 、 PRIV_DACWRITE 、 PRIV_LIMIT、PRIV_OWNER**
msgget()	**PRIV_COMMALLOWED**
msgrcv()	**PRIV_COMMALLOWED、PRIV_DACREAD**
msgsnd()	**PRIV_COMMALLOWED、PRIV_DACWRITE**
munlock()	**PRIV_MLOCK**
munlockall()	**PRIV_MLOCK**
nice()	**PRIV_COMMALLOWED、PRIV_LIMIT、PRIV_OWNER**
open()	**PRIV_CMPTREAD 、 PRIV_CMPTWRITE 、 PRIV_DACREAD 、 PRIV_DACWRITE、PRIV_LIMIT**

pipe()	**PRIV_LIMIT**
plock()	**PRIV_MLOCK**
pset_assign()	**PRIV_PSET**、**PRIV_RTPSET**
pset_bind()	**PRIV_PSET**、**PRIV_RTPSET**
pset_create()	**PRIV_PSET**、**PRIV_RTPSET**
pset_ctl()	**PRIV_PSET**、**PRIV_RTPSET**
pset_destroy()	**PRIV_PSET**、**PRIV_RTPSET**
pset_getattr()	**PRIV_PSET**、**PRIV_RTPSET**
pset_setattr()	**PRIV_PSET**、**PRIV_RTPSET**
pstat()	**PRIV_COMMALLOWED**、[**PRIV_OWNER**]；有关详细信息，请参阅“用于 pstat 系统调用的权限”。
ptrace()	**PRIV_COMMALLOWED**、**PRIV_OWNER**
quotactl()	**PRIV_CMPTREAD**、**PRIV_DACREAD**、**PRIV_SYSATTR**
readlink()	**PRIV_CMPTREAD**、**PRIV_DACREAD**
reboot()	**PRIV_REBOOT**
rename()	**PRIV_CMPTREAD**、**PRIV_CMPTWRITE**、**PRIV_DACREAD**、**PRIV_DACWRITE**、**PRIV_OWNER**
rmdir()	**PRIV_CMPTREAD**、**PRIV_CMPTWRITE**、**PRIV_DACREAD**、**PRIV_DACWRITE**、**PRIV_OWNER**
rtprio()	**PRIV_COMMALLOWED**、**PRIV_OWNER**、**PRIV_RTPRIO**
sched_getparam()	**PRIV_COMMALLOWED**
sched_getscheduler()	**PRIV_COMMALLOWED**
sched_rr_get_interval()	**PRIV_COMMALLOWED**
sched_setparam()	**PRIV_COMMALLOWED**、**PRIV_OWNER**、**PRIV_RTSCHED**
sched_setscheduler()	**PRIV_COMMALLOWED**、**PRIV_OWNER**、**PRIV_RTSCHED**
sem_open()	**PCIV_COMMALLOWED**、**PRIV_DACREAD**、**PRIV_DACWRITE**
sem_unlink()	**PRIV_COMMALLOWED**、**PRIV_DACWRITE**
semctl()	**PRIV_COMMALLOWED**、**PRIV_DACREAD**、**PRIV_DACWRITE**、**PRIV_OWNER**

semget()	**PRIV_COMMALLOWED**
semop()	**PRIV_DACREAD**、**PRIV_DACWRITE**、**PRIV_COMMALLOWED**
semtimedop()	**PRIV_DACREAD**、**PRIV_DACWRITE**、**PRIV_COMMALLOWED**
serialize()	**PRIV_SERIALIZE**
setacl()	**PRIV_CMPTREAD**、**PRIV_DACREAD**
setaudid()	**PRIV_SELFAUDIT**
setaudproc()	**PRIV_SELFAUDIT**
setdomainname()	**PRIV_SYSATTR**
setevent()	**PRIV_AUDCONTROL**
setgid()	**PRIV_CHSUBJIDENT**
setgroups()	**PRIV_CHSUBJIDENT**
sethostname()	**PRIV_SYSATTR**
setpgrp()	**PRIV_SESSION**
setpgrp2()	**PRIV_COMMALLOWED**
setpriority()	**PRIV_COMMALLOWED**、**PRIV_LIMIT**、**PRIV_OWNER**
setprivgrp()	**PRIV_SYSATTR**
setregid()	**PRIV_CHSUBJIDENT**
setresgid()	**PRIV_CHSUBJIDENT**
setresuid()	**PRIV_CHSUBJIDENT**
setrlimit()	**PRIV_LIMIT**
setsid()	**PRIV_SESSION**
setsockopt()	**PRIV_NETBROADCAST**：根据使用的选项而有所不同。
settimeofday()	**PRIV_SYSATTR**
settune()	**PRIV_SYSATTR**
settune_txn()	**PRIV_SYSATTR**
setuid()	**PRIV_CHSUBJIDENT**
setuname()	**PRIV_SYSATTR**

shm_open()	**PRIV_CMPTREAD、PRIV_CMPTWRITE、PRIV_DACREAD、PRIV_DACWRITE**
shm_unlink()	**PRIV_CMPTWRITE、PRIV_DACWRITE、PRIV_OWNER**
shmat()	**PRIV_COMMALLOWED、PRIV_DACREAD、PRIV_DACWRITE**
shmctl()	**PRIV_COMMALLOWED、PRIV_COMMALLOWED、PRIV_DACREAD、PRIV_MLOCK、PRIV_OWNER**
shmget()	**PRIV_COMMALLOWED**
sigqueue()	**PRIV_COMMALLOWED、PRIV_OWNER**
socket()	**PRIV_LIMIT**
socketpair()	**PRIV_LIMIT**
stat()	**PRIV_CMPTREAD、PRIV_DACREAD**
statfs()	**PRIV_CMPTREAD、PRIV_DACREAD**
statfvfs()	**PRIV_CMPTREAD、PRIV_DACREAD**
stime()	**PRIV_SYSATTR**
swapon()	**PRIV_MOUNT**
symlink()	**PRIV_CMPTREAD、PRIV_CMPTWRITE、PRIV_DACREAD、PRIV_DACWRITE、PRIV_LIMIT**
truncate()	**PRIV_CMPTREAD、PRIV_CMPTWRITE、PRIV_DACREAD、PRIV_DACWRITE、PRIV_OBJSUID**、PRIV_OWNER
ttrace()	**PRIV_COMMALLOWED、PRIV_OWNER**
ulimit()	**PRIV_LIMIT**
umount()	**PRIV_MOUNT、PRIV_OWNER**
unlink()	**PRIV_CMPTREAD、PRIV_CMPTWRITE、PRIV_DACREAD、PRIV_DACWRITE、PRIV_FSINTEGRITY、PRIV_OWNER**
ustat()	**PRIV_SYSATTR**
utime()	**PRIV_OWNER**
vfsmount()	**PRIV_MOUNT**
write()	**PRIV_LIMIT**

警告

如上所述，产品文档描述了程序或用户可以获得足够的权限执行受限的操作的替代方式。

网络问题

权限没有在分布式系统中扩展。它们仅适用于在本地系统上使用。例如，如果必须覆盖任意的限制来访问另一个系统上的文件，则拥有 **PRIV_DACREAD** 或 **PRIV_DACWRITE** 权限的进程便不能执行这一操作。

再如，如果系统的 NFS 子系统配置为可将用户 ID 零转换为用户 ID **UID_NOBODY** ，它仍将如此操作。此外，一些系统守护程序通过检查连接是否源自某个权限端口（通常为 **0-1023** ）来确定是允许还是禁止该连接。不可也不应当更改此行为。

权限升级

在某些情况下，可以将单个权限或一组权限加载到未被明确授予其他权限的进程。这称之为权限升级。

例如，只拥有 **PRIV_DACWRITE** 权限的用户或许仅能覆盖关键的操作系统文件，而在进程中，可能会向它自己授予除 **PRIV_DACWRITE** 之外的其他权限。

另请参阅

crontab(1)、 sam(1M)、 setfilexsec(1M)、 setrules(1M)、 shutdown(1M)、 acct(2)、 audwrite(2)、 execve(2)、 getfh(2)、 mknod(2)、 modload(2)、 modpath(2)、 modstat(2)、 mount(2)、 nice(2)、 setrlimit(2)、 priv_add_effective(3)、 priv_remove(3)、 privileges(3)、 compartments(4)、 compartments(5)、 privgrp(5)、 glossary(9)。

名称

process_id_max - 限制进程 ID (PID) 的最大值

值

保证安全

30000

缺省值

30000

最小值

process_id_min + 512

最大值

1,073,741,823

process_id_max 必须大于或等于 **process_id_min + 512** 。如果 **process_id_max** 和 **process_id_min** （包括边界值）之间的差值小于 **nproc** ，则会有效地将 **nproc** 限制到此差值内。

说明

process_id_max 可调参数允许管理员为 **fork** （请参阅 *fork*(2) ）生成的进程 ID (PID) 选择可能值的范围。管理员可以利用它在兼容性、容量和审美方面进行权衡。

警告：有些程序无法接受 PID 值达到最大值。如果存在此类程序且是关键程序，则应适当限制最大 PID。有关这些内容的详细信息，请参阅下面的“可能的应用程序问题”一节。

更改此可调参数的人员

任何用户。

更改限制

如果存在关键应用程序且假定 PID 适合受限范围，请不要增加最大 PID（请参阅下面的“警告”一节）。

可以随时增加 **process_id_max** 的值，它会立即生效（但是，在创建足够数量的新进程以便使系统利用可用的更高值之前，系统不会立即通知生效）。

减少 **process_id_max** 的值也可以立即生效。但是，PID 大于新值的现有进程都不会受到影响。只有在重新引导后，PID 减少的所有进程才会完全生效。

应该何时增加此可调参数的值

如果需要增加 **process_id_min** 和 **process_id_max** 可调参数定义的 PID 范围，以便能创建更多并发进程，则请增加最大 PID。有关进程的数量限制，请参阅 *nproc*(5) 。

增加含有许多活动进程（例如，大于 25,000）的系统中的最大 PID。更大的范围可能会增加创建新进程的效率（因为它需要花费较少的工作即可找到可用的 PID）。

如果需要验证软件程序在 PID 值较大的环境中是否能正常执行，则增加 **process_id_max** 可调参数以及

process_id_min 可调参数，以强制所有新进程 ID 都采用较大值（有关详细信息，请参阅 *process_id_min*(5)）。

如果存在关键应用程序且假定 PID 适合受限制范围，请不要增加最大 PID（请参阅下面的“警告”一节）。

增加此值的负面影响

如果 **process_id_max** 和 **process_id_min** 之间的差值小于 **nproc**，则将允许同时存在的进程数限制为此差值。

应该何时降低此可调参数的值

如果关键应用程序假定 PID 范围是受限制的，则降低最大 PID。

降低此值的负面影响

如果 **process_id_max** 和 **process_id_min** 之间的差值小于 **nproc**，则将允许同时存在的进程数限制为此差值。

应该同时更改的其他可调参数值

可能需要更改 **nproc**。对于程序开发和验证，可能还需要更改 **process_id_min** 可调参数。

可能的应用程序问题

以前，PID 值的范围限制为 0 至 30,000。某些程序对此范围进行内置假定。本节简要介绍其中一些假定。

某些应用程序内置假定 PID 不超过 30,000（这是 **MAXPID**（未列入正式文件）和 **PID_MAX** 常量的以前的值）。如果 PID 超过此最大值，则这些程序可能失败。

某些应用程序在 16 位变量（C 中的类型 **short**）中存储 PID。如果最大 PID 超过 32,767，则此类程序可能失败。

某些程序提供了区分 PID 位数的输出格式。如果 PID 超过 5 位（超过 99,999），则此类程序生成的输出可能会不美观。在某些情况下，输出字段的自动扩展会影响列的对齐方式。而在其他一些情况下，相邻字段可能会连在一起，使输出难于理解。

某些程序或脚本会解析包含 PID 值的其他程序的输出。某些此类程序内置假定 PID 将不会超过五个字符位置。如果范围超过 99,999，则此类程序可能会失败。

因为会话 ID (SID) 和进程组 ID (PGID) 与此会话或组创建者的进程 ID 相同，所以增加最大 PID 还会增加最大 SID 和 PGID。尽管 SID 和 PGID 很不相同，但也存在相同的应用程序问题。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

为了满足 PID 生成算法的需要，HP-UX 内核可能会以无提示方式舍入为 **process_id_max** 和（或） **process_id_min** 选择的值（例如，最接近的 2 的幂）。

如果存在关键应用程序且假定 PID 适合受限范围，请不要增加最大 PID。有关此类编程假定的详细信息，请参阅上一节“可能的应用程序问题”。系统已选择缺省的最大值 (30,000)，以便提供与所有此类程序的兼容性。如果

程序对较大 PID 值的敏感度未知，则应使用此值。有关要为软件验证选择多大 PID 值的信息，请参阅 *process_id_min*(5) 。

增加 PID 范围不会增加系统中进程的最大数。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

process_id_max 由 HP 开发。

另请参阅

fork(2)、 nproc(5)、 process_id_min(5)。

《Number of Processes and Process ID Values on HP-UX》白皮书，可从 **http://docs.hp.com** 上获得。

名称

process_id_min - 指定进程 ID (PID) 的最小值

值

保证安全

0

缺省值

0

最小值

0

最大值

process_id_max - 512

process_id_min 必须小于或等于 **process_id_max - 512** 。如果 **process_id_max** 和 **process_id_min** （含边界值）之间的差值小于 **nproc** ，则会有效地将 **nproc** 限制到此差值内。

说明

process_id_min 可调参数指定要为新进程生成的最小进程 ID (PID) 值。它允许应用程序开发人员复制具有较大 PID 值的环境，以进行软件验证。

此可调参数主要用于开发环境。

更改此可调参数的人员

关注具有较大 PID 值的环境中的软件程序验证的任何人。

更改限制

对此可调参数的更改会立即生效。不过，这并不会更改现有进程及其 PID。

建议在指定此可调参数后重新引导系统，这样系统中的所有 PID 都会具有所选范围中的值。

应该何时增加此可调参数的值

增加此可调参数的值，以验证软件是否接受开发环境中具有较大的 PID 值。

增加此值的负面影响

如果 **process_id_max** 和 **process_id_min** 之间的差值小于 **nproc** ，则允许同时存在的进程数将低于 **nproc** 的值。

应该何时降低此可调参数的值

当系统要从开发环境移动到生产环境时，可降低此可调参数的值。

降低此值的负面影响

如果 **process_id_max** 和 **process_id_min** 之间的差值小于 **nproc** ，则允许同时存在的进程数将低于 **nproc** 的值。

应该同时更改的其他可调参数值

可能需要更改 **nproc** 。理想情况下，应将 **process_id_max** 可调参数设置为验证使用 PID 的软件时的最大值。更改 **process_id_min** 后，建议重新引导系统，以确保所有进程都具有所需范围中的 PID。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

为了满足 PID 生成算法的需要，HP-UX 内核可能会以无提示方式舍入为 **process_id_max** 和（或） **process_id_min** 选择的值（例如，最接近的 2 的幂）。

建议以 **process_id_max** 的可调参数的最大可配置值 (1,073,741,567) 以及设置为同级别值（如 1,000,000,000）的 **process_id_min** 的来执行软件验证。

有关使用较大 PID 值所带来的可能应用程序问题的信息，请查阅 **process_id_max** （请参阅 *process_id_max*(5) ）的联机帮助页。

因为会话 ID (SID) 和进程组 ID (PGID) 与此会话或组创建者的进程 ID 相同，所以增加最小 PID 还会增加最小 SID 和 PGID。

系统可能会为某些特定的系统进程分配小于 **process_id_min** 的进程 ID 值。

在 **process_id_min** 设置为非零值的引导系统可能会使初始化进程（又称为 init）具有除 1 外的 PID。不过，缺省情况下，大多数系统实用程序和命令都会将初始化进程 PID 的值报告为 1。例如，如果调用方的父进程是初始化进程，则 **getppid()** 将返回 1，而不论实际的 PID 是多少。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

process_id_min 由 HP 开发。

另请参阅

init(1M)、 fork(2)、 getppid(2)、 nproc(5)、 process_id_max(5)。

《Number of Processes and Process ID Values on HP-UX》白皮书，可从 **http://docs.hp.com** 上获得。

名称

pthread_scope_options - 指定线程调度争用范围的外部选项列表

说明

HP-UX 11i v2 及之前的版本中，HP-UX 支持“MxN”线程模型。在 MxN 模型下，应用程序可以灵活选择应用程序中的线程类型（取决于指定的争用范围）。基于争用范围有两种线程类型，即 **PTHREAD_SCOPE_SYSTEM** 线程及 **PTHREAD_SCOPE_PROCESS** 线程。

pthread_attr_init() 可用于初始化属性对象。随后， **pthread_attr_setscope()** 可用于显式设置范围。如果不使用 **pthread_attr_setscope()** ，则视为缺省争用范围。

缺省争用范围将为 **PTHREAD_SCOPE_SYSTEM** 。

提供了一些外部选项以控制争用范围，而无须修改应用程序源。这在应用程序开发者为特定的应用程序决定合适的线程模型之前，以多个线程模型来检查应用程序性能的过程中是非常有用的。

有三种不同的方式指定外部范围选项:

1. 编译时。
2. 属性文件中。
3. 环境变量。

1. 编译时选项

编译时提供的选项为:

PTHREAD_FORCE_SCOPE_SYSTEM
: 强制设置系统范围，与属性中为线程创建指定的范围无关。

PTHREAD_FORCE_SCOPE_PROCESS
: 强制设置进程范围，与属性中为线程创建指定的范围无关。

PTHREAD_DEFAULT_SCOPE_PROCESS
: 如果争用范围由 **pthread_attr_setscope()** 设置，那么该范围被视为创建线程时的范围。否则，视 **PTHREAD_SCOPE_PROCESS** 为创建线程时的范围。

可以通过 **-D** 指定编译时选项或之前通过定义包括应用程序来源文件 **pthread.h** 的编译时选项。如果没有指定上述选项，除非 **pthread_attr_setscope()** 指定了不同的选项，才会创建系统范围线程。

PTHREAD_COMPAT_MODE 编译选项与 **PTHREAD_FORCE_SCOPE_SYSTEM** 相似，提供它是为了向后兼容。然而，如果使用了上述任何编译选项， **PTHREAD_COMPAT_MODE** 将不会产生任何影响。

2. 属性文件选项

用户可以灵活选择使用 **libpthread** 属性文件为库指定各种可调参数，帮助优化其应用程序，而不用更改任何源代码。属性文件缺省路径名称是 **/usr/lib/libpthread.properties** 。请注意，该文件将必须由用户显式创建。如果应用程序需要 pthread 库读取属性文件中的可调参数，环境变量 **PTHREAD_TUNE** 必须被设置为 1、on 或 ON。用户

可以通过环境变量 **PTHREAD_PROPERTY_FILE** 为属性文件指定不同的位置。

属性文件中，以 **#** 开始的行是注释。对非注释行，第一个和第二个单词会被解压缩。第一个单词应该是其中一个可调参数的名称，第二个单词应该是该变量的值。

属性文件支持下列可调参数来从外部控制线程范围：

force_scope_system
强制设置系统范围，与属性中为线程创建指定的范围无关。

force_scope_process
强制设置进程范围，与属性中为线程创建指定的范围无关。

default_scope_process
如果争用范围由 **pthread_attr_setscope()** 设置，那么该范围被视为创建线程时的范围。否则，视 **PTHREAD_SCOPE_PROCESS** 为创建线程时的范围。

default_scope_system
如果争用范围由 **pthread_attr_setscope()** 设置，那么该范围被视为创建线程时的范围。否则，视 **PTHREAD_SCOPE_SYSTEM** 为创建线程时的范围。

注释：提供 HP-UX 11i v2 支持的属性文件选项 **mxnfromcompatmode** 和 **all1x1** 是为了向后兼容。然而，如果使用了上述任何编译选项， **mxnfromcompatmode** 和 **all1x1** 将不会产生任何影响。

3. 环境变量

有一组环境变量，设置后可以覆盖应用程序中指定的线程范围。应该在开始运行应用程序之前设置好这些环境变量。

支持控制范围的环境变量为：

PTHREAD_FORCE_SCOPE_SYSTEM
强制设置系统范围，与属性中为线程创建指定的范围无关。

PTHREAD_FORCE_SCOPE_PROCESS
强制设置进程范围，与属性中为线程创建指定的范围无关。

PTHREAD_DEFAULT_SCOPE_PROCESS
如果争用范围由 **pthread_attr_setscope()** 设置，那么该范围被视为创建线程时的范围。否则，视 **PTHREAD_SCOPE_PROCESS** 为创建线程时的范围。

PTHREAD_DEFAULT_SCOPE_SYSTEM
如果争用范围由 **pthread_attr_setscope()** 设置，那么该范围被视为创建线程时的范围。否则，视 **PTHREAD_SCOPE_SYSTEM** 为创建线程时的范围。

注释：HP-UX 11i v2 支持的 **PTHREAD_COMPAT_MODE** 环境变量仍将得到支持，并且具有与 **PTHREAD_FORCE_SCOPE_SYSTEM** 相同的效果。然而，如果设置了上述任何环境变量， **PTHREAD_COMPAT_MODE** 将不会产生任何影响。

多个选项指定时的优先级：

- 环境变量设置优先于属性文件设置，而属性文件设置又依次优先于编译时选项。
- 如果上述指定的其中一种方式不只进行一项设置，那么强制选项优先于缺省选项和 **pthread_attr_setscope()** 设置的范围。系统范围优先于进程范围。 **pthread_attr_setscope()** 设置的争用范围优先于为缺省范围进程和缺省范围系统设置的外部范围。

举例

1. 下列设置在属性文件中完成：

   force_scope_system 1

   default_scope_process 1

   export PTHREAD_TUNE=1 和
   export PTHREAD_PROPERTY_FILE=*user's_properties_file*

   应该为 pthread 库读取的属性文件而完成。

   如果未设置 **PTHREAD_PROPERTY_FILE** ，则将使用 **/usr/lib/libpthread.properties** 。

   使用这些设置， **force_scope_system** 拥有优先级，且将为所有的线程创建强制设置系统范围。

2. 如果编译时由 **-D** 或是其中一个应用程序头文件对下列选项进行了定义：**PTHREAD_DEFAULT_SCOPE_SYSTEM** 及 **PTHREAD_FORCE_SCOPE_SYSTEM** ；

   即使用 **pthread_attr_setscope()** 指定进程范围， **PTHREAD_FORCE_SCOPE_SYSTEM** 也具有更高的优先级，同时会强制执行系统范围线程。

3. 如果设置了下列环境变量（假定使用 sh-posix）：

   export PTHREAD_DEFAULT_SCOPE_PROCESS=1

   export PTHREAD_FORCE_SCOPE_SYSTEM=ON

   PTHREAD_FORCE_SCOPE_SYSTEM 将具有更高优先级，同时为进程中创建的所有线程强制设置系统范围。

4. 如果设置了下列环境变量（假定使用 sh-posix）：

   export PTHREAD_DEFAULT_SCOPE_SYSTEM=on

   export PTHREAD_FORCE_SCOPE_PROCESS=ON

   PTHREAD_FORCE_SCOPE_PROCESS 将具有优先级，同时为进程中创建的所有线程强制设置进程范围。

另请参阅

pthread(3T)、pthread_attr_init(3T)、 pthread_attr_setscope(3T)。

名称

pthread_stubs - 在 C 库中为其提供存根的 pthread 调用列表

说明

libc 累积修补软件 **PHCO_22923** (11.00) 和 **PHCO_23772** (11.11) 以前版本中的 libc 共享库，包含在 **libpthread** 和 **libcma** 中 pthread 函数的存根。存根允许非线程应用程序成功地动态加载线程安全库以解析 pthread 符号。必须在链接行上没有 **-lpthread** 或 **-lcma** 的情况下构建对存根的 pthread/cma 调用进行解析的应用程序。除了 *pthread_getspecific*(3T) 系列 API 在存根中实现完整的功能外，**libc** 提供的存根不具备任何功能，都是返回零的哑函数。

对下面的任何存根函数的 pthread 调用均返回零。

pthread_atfork(3T)
pthread_attr_destroy(3T)
pthread_attr_getdetachstate(3T)
pthread_attr_getguardsize(3T)
pthread_attr_getinheritsched(3T)
pthread_attr_getschedparam(3T)
pthread_attr_getschedpolicy(3T)
pthread_attr_getscope(3T)
pthread_attr_getstackaddr(3T)
pthread_attr_getstacksize(3T)
pthread_attr_setdetachstate(3T)
pthread_attr_setguardsize(3T)
pthread_attr_setinheritsched(3T)
pthread_attr_setschedparam(3T)
pthread_attr_setschedpolicy(3T)
pthread_attr_setscope(3T)
pthread_attr_setstackaddr(3T)
pthread_attr_setstacksize(3T)
pthread_cancel(3T)
pthread_cond_broadcast(3T)
pthread_cond_destroy(3T)
pthread_cond_init(3T)
pthread_cond_signal(3T)
pthread_cond_timedwait(3T)
pthread_cond_wait(3T)
pthread_condattr_destroy(3T)
pthread_condattr_getpshared(3T)
pthread_condattr_init(3T)
pthread_condattr_setpshared(3T)

pthread_continue(3T)
pthread_detach(3T)
pthread_getconcurrency(3T)
pthread_getschedparam(3T)
pthread_join(3T)
pthread_kill(3T)
pthread_mutex_destroy(3T)
pthread_mutex_getprioceiling(3T)
pthread_mutex_init(3T)
pthread_mutex_lock(3T)
pthread_mutex_setprioceiling(3T)
pthread_mutex_trylock(3T)
pthread_mutex_unlock(3T)
pthread_mutexattr_destroy(3T)
pthread_mutexattr_getprioceiling(3T)
pthread_mutexattr_getprotocol(3T)
pthread_mutexattr_getpshared(3T)
pthread_mutexattr_gettype(3T)
pthread_mutexattr_init(3T)
pthread_mutexattr_setprioceiling(3T)
pthread_mutexattr_setprotocol(3T)
pthread_mutexattr_setpshared(3T)
pthread_mutexattr_settype(3T)
pthread_once(3T)
pthread_rwlock_destroy(3T)
pthread_rwlock_init(3T)
pthread_rwlock_rdlock(3T)
pthread_rwlock_tryrdlock(3T)
pthread_rwlock_trywrlock(3T)
pthread_rwlock_unlock(3T)
pthread_rwlock_wrlock(3T)
pthread_rwlockattr_destroy(3T)
pthread_rwlockattr_getpshared(3T)
pthread_rwlockattr_init(3T)
pthread_rwlockattr_setpshared(3T)
pthread_self(3T)
pthread_setcancelstate(3T)
pthread_setcanceltype(3T)
pthread_setconcurrency(3T)

pthread_setschedparam(3T)
pthread_sigmask(3T)
pthread_suspend(3T)
pthread_testcancel(3T)

下列 pthread 调用的存根具有完整的功能。有关详细信息，请参考 *pthread*(3T) 。

pthread_key_create(3T)
pthread_getspecific(3T)
pthread_setspecific(3T)
pthread_key_delete(3T)
pthread_exit(3T)

调用下列存根。

pthread_self(3T) 始终返回 1。
pthread_equal(*arg1,arg2***)** 返回 (*arg1*==*arg2*)。
pthread_create(3T) 和 *pthread_attr_init*(3T) 返回 [ENOSYS] 。

上述存根在 libc 中提供，因为在 HP-UX 中，如果非线程应用程序链接至线程安全库，那么从应用程序中调用线程安全例行程序在运行时将会由于未解析的 **pthread_*** 形式的符号而失败。为了解析这些符号，有必要将非线程应用程序链接至线程库 （**libpthread** 或 **libcma** ）。然而，链接至线程库会强制应用程序在即使没有创建线程的情况下使用线程安全功能，这样会导致后续的性能丢失。

为了克服上述问题，C 库已经为 **POSIX.1c** API 提供了存根。在 HP-UX C 语言库中为 **POSIX.1c** API 提供存根对非线程应用程序具有两种直接影响:

- 如果非线程应用程序链接至线程安全库，则将解析 **POSIX.1c** 线程符号。
- 避免实际线程库的开销。尤其当非线程应用程序使用线程存根而非实际线程库过程时，可以避免与互斥体关联的开销。

链接顺序问题

当应用程序有意使用实际 pthread API 或 cma API 时，由于链接顺序问题，它可能在无意中提取到当前在 **libc** 中的存根。需要 cma 行为的应用程序必须链接至 **libcma** ，并且必须以支持的链接顺序执行此操作；也就是说，链接行只应是共享的，且在 **-lcma** 之前不应包含 **-lc** 。只要满足了此条件，就可以引用正确的 cma 函数。同样，需要 pthread 库行为的多线程应用程序必须链接至 libpthread，并且必须以支持的链接顺序执行此操作，同时只使用共享的 **libc** 和 **libpthread** 。

举例

以下示例为可能存在的链接顺序问题。

示例 1

对于需要 pthread/cma 调用解析至 **libc** 中的 pthread 存根的应用程序或任何链接库，必须在链接行上没有 **-lpthread** 或 **-lcma** 的情况下构建。

如果在链接行上于 **-lpthread** 或 **-lcma** 之前指定了 **-lc**，则 pthread/cma 调用解析至 **libc** 中的 pthread 存根。这可能导致以下示例给出的问题:

```
$ cat thread.c
#include <pthread.h>
#include <stdio.h>

void *thread_nothing(void *p)
{
printf("Success\n");
}

int main()
{
int err;
pthread_t thrid;

err = pthread_create(&thrid, (pthread_attr_t *) NULL, thread_nothing,
(void *) NULL);
sleep(1);
if (err)
{
printf("Error\n");
return err;
}
}

$ cc thread.c -lc -lpthread
$ a.out
Error
$ chatr a.out
a.out:
shared executable
shared library dynamic path search:
SHLIB_PATH disabled second
embedded path disabled first Not Defined
shared library list:
dynamic /usr/lib/libc.2 <- libc before libpthread
dynamic /usr/lib/libpthread.1
shared library binding:
```

deferred
global hash table disabled ...

示例 **1** 的解决方案

对线程应用程序，将环境变量 **LD_PRELOAD** 设置为 **libpthread** 库来运行可执行文件，或将可执行文件与 **-lpthread** 链接：

```
$ LD_PRELOAD="/usr/lib/libpthread.1" a.out
Success

$ cc thread.c -lpthread
$ a.out
Success
$ chatr a.out
a.out:
shared executable
shared library dynamic path search:
SHLIB_PATH disabled second
embedded path disabled first Not Defined
shared library list:
dynamic /usr/lib/libpthread.1
dynamic /usr/lib/libc.2
shared library binding:
deferred
global hash table disabled ...
```

示例 **2**

在线程应用程序中于 **-lpthread** 之前指定 **-lc** 可能导致如下运行时问题，因为 pthread 调用被解析至 **libc** 中的存根，而不是 pthread 库中的函数。

- 由于未初始化的内部结构，对 pthread 函数的调用失败。
- 调用 *gethostbyname*(3N) 失败并返回空值。
- Apache webmin 和 perl DBI 应用程序失败，同时显示下列错误消息：

  Can't load *libname***.sl for module** *xxx***: Invalid argument at** *address*

- 调用 *shl_load*(3X) 失败，同时显示下列错误消息：

errno 22 (invalid argument)

因为 **pthread_mutex_lock** 存根返回零。

```
$ cat a.c
#include <stdio.h>
#include <dl.h>

extern int errno;

main()
{
shl_load("lib_not_found", BIND_DEFERRED, 0);
printf("Error %d, %s\n", errno, strerror(errno));
}
$ cc a.c -lc -lpthread
$ a.out
Error 22, Invalid argument
$ LD_PRELOAD=/usr/lib/libpthread.1 ./a.out
Error 2, No such file or directory

$ cat b.c
#include <stdio.h>
#include <dlfcn.h>

void* handle;
extern int errno;

main()
{
handle = dlopen("lib_not_found", RTLD_LAZY);
printf("Error %d, %s\n", errno, strerror(errno));
if (handle == NULL)
{
printf("Error: %s\n",dlerror());
}
}
$ cc b.c -lc -lpthread
$ a.out
Error 22, Invalid argument
```

```
Error:
$ ./a.out
$ LD_PRELOAD=/usr/lib/libpthread.1
Error 0, Error 0
Error: Can't open shared library: lib_not_found
```

由于上述问题，无论在何种情况下，在构建可执行文件或共享库的命令中都不应该指定 **-lc** 。缺省情况下，编译程序驱动程序（ cc 、 **aCC** 、 **f90** ）自动将 **-lc** 传递至可执行文件链接行结尾处的链接程序。要了解共享库是否使用 **-lc** 构建，请查看 **chatr** 输出中的共享库列表（请参阅 *chatr*(1) ），或使用 **ldd** 列出相关的库（请参阅 *ldd*(1)）：

```
$ cc +z -c lib1.c

$ ld -b -o lib1.sl lib1.o -lc

$ ldd lib1.sl
/usr/lib/libc.2 => /usr/lib/libc.2
/usr/lib/libdld.2 => /usr/lib/libdld.2
/usr/lib/libc.2 => /usr/lib/libc.2

$ cc +DA2.0W +z -c lib1.c

$ ld -b -o lib1.sl lib1.o -lc

$ ldd lib1.sl
libc.2 => /lib/pa20_64/libc.2
libdl.1 => /usr/lib/pa20_64/libdl.1
```

要查看运行时加载的相关共享库的顺序（顺序只在 64 位模式时有效），请对可执行文件上使用 **ldd** 命令（32 位模式的 ldd 将按相反顺序显示加载库的顺序）：

```
$ cc +DA2.0W thread.c -lpthread

$ ldd a.out
libpthread.1 => /usr/lib/pa20_64/libpthread.1
libc.2 => /usr/lib/pa20_64/libc.2
libdl.1 => /usr/lib/pa20_64/libdl.1
$ cc +DA2.0W thread.c -lc -lpthread

$ ldd a.out
libc.2 => /usr/lib/pa20_64/libc.2
```

```
libpthread.1 => /usr/lib/pa20_64/libpthread.1
libdl.1 => /usr/lib/pa20_64/libdl.1

$ cc +DA2.0W thread.c -lpthread -lc

$ ldd a.out
libpthread.1 => /usr/lib/pa20_64/libpthread.1
libc.2 => /usr/lib/pa20_64/libc.2
libdl.1 => /usr/lib/pa20_64/libdl.1
```

建议:

- 删除所有共享库构建命令中的 **-lc**
- 删除所有可执行文件构建命令中的 **-lc**
- 将 **LD_PRELOAD** 环境变量设置为 **libpthread** 或 **libcma** 的完整路径名，这将导致在程序启动时在其他相关库之前加载该库。在 **PHSS_22478** 以及更高版本的链接程序修补软件中可以使用 **LD_PRELOAD** 功能。请参阅 *dld.sl*(5) 联机帮助页。
- 如果直接使用 *ld*(1) 命令而不是使用编译程序驱动程序链接，那么将 **-lc** 作为链接行的最后部分添加。

示例 **3**（**64** 位）

如果一个 64 位的共享库是使用 **-lpthread** 构建的，但可执行文件不是，则会在 **libpthread** 之前加载 **libc** （由于按宽度优先搜索），同时 pthread 调用被解析至 libc 中的 pthread 存根。运行时，当 **a.out** 加载后，按宽度优先顺序加载 **a.out** 的相关对象：在 **libpthread** 作为 **libc.2** 的相关对象加载之前，会先将 **libc** 作为 **a.out** 的相关对象加载。第一种情况的相关性列表为:

```
     a.out
    /   /    \
lib1 lib2 libc
 |     |
libc libpthread
```

因此构建加载图形如下:

lib1.sl --> lib2.sl -->libc.2 --> libpthread.1

这是非线程应用程序所需的行为，但会导致线程应用程序（使用 **libpthread** 或 **libcma** ）失败。

lib1.sl
指定 **-lc** ， **lib2.sl** 指定 **-lpthread** 且 **a.out** 上没有 **-lpthread** 。

```
$ cc -c +z +DA2.0W lib1.c lib2.c
lib1.c:
lib2.c:
```

```
$ ld -b -o lib1.sl -lc lib1.o

$ ld -b -o lib2.sl -lpthread lib2.o

$ cc +DA2.0W thread.c -L. -l1 -l2

$ a.out
Error

$ ldd a.out
lib1.sl => ./lib1.sl
lib2.sl => ./lib2.sl
libc.2 => /usr/lib/pa20_64/libc.2
libc.2 => /lib/pa20_64/libc.2
libpthread.1 => /lib/pa20_64/libpthread.1
libdl.1 => /usr/lib/pa20_64/libdl.1
```

lib2.sl 指定 **-lpthread** 且 **a.out** 上没有 **-lpthread** 。

```
$ ld -b -o lib1.sl lib1.o

$ ld -b -o lib2.sl -lpthread lib2.o

$ cc +DA2.0W thread.c -L. -l1 -l2

$ a.out
Error

$ ldd a.out
lib1.sl => ./lib1.sl
lib2.sl => ./lib2.sl
libc.2 => /usr/lib/pa20_64/libc.2
libpthread.1 => /lib/pa20_64/libpthread.1
libdl.1 => /usr/lib/pa20_64/libdl.1
```

如果 **libcma** 被列为共享库的相关库，也会发生同样的问题，需要将可执行文件与 **-lcma** 链接。

示例 3 的建议

对线程应用程序，将 **LD_PRELOAD** 设置为 **libpthread** 库来运行可执行文件，或将可执行文件与 **-lpthread** 链接：

首先使用 **LD_PRELOAD** 加载 **libpthread**

```
$ ld -b -o lib1.sl lib1.o

$ ld -b -o lib2.sl -lpthread lib2.o

$ cc +DA2.0W thread.c -L. -l1 -l2

$ a.out
Error

$ ldd a.out
lib1.sl => ./lib1.sl
lib2.sl => ./lib2.sl
libc.2 => /usr/lib/pa20_64/libc.2
libpthread.1 => /lib/pa20_64/libpthread.1
libdl.1 => /usr/lib/pa20_64/libdl.1

$ LD_PRELOAD="/lib/pa20_64/libpthread.1" a.out
Success
```

a.out 将正确列出线程应用程序的 **-lpthread** 。

```
$ ld -b -o lib1.sl lib1.o

$ ld -b -o lib2.sl -lpthread lib2.o

$ cc +DA2.0W thread.c -L. -l1 -l2 -lpthread

$ a.out
Success

$ ldd a.out
lib1.sl => ./lib1.sl
lib2.sl => ./lib2.sl
libpthread.1 => /usr/lib/pa20_64/libpthread.1
libc.2 => /usr/lib/pa20_64/libc.2
libpthread.1 => /lib/pa20_64/libpthread.1
libdl.1 => /usr/lib/pa20_64/libdl.1
```

示例 **4**（已归档的 **libc**）

如果共享库中的链接行包含 **-lc** 以显式链接至 **libc** 中，则请删除 **-lc** 。否则，共享库可能引用 **libc.2** 而 **a.out** 可能引用较旧的（已归档的） **libc** 版本。因此应用程序实际上会使用两种不同的 **libc** 版本且可能混淆代码。这可能导

致兼容性问题。基本上，应用程序或库不论在何种情况下都不应该直接链接 **libc** 。所有程序均需链接 **libc** （编译程序自动链接），因此共享库将始终具有正确执行所需的接口而不必在链接行上指定 **-lc** 。

另请参阅

chatr(1)、 ld(1)、 ldd(1)、 pthread(3T)、 shl_load(3X)、 dld.sl(5)。

名称

quota - 磁盘配额

说明

系统管理员可以使用“磁盘配额”，按文件系统来限制用户或组可拥有的文件和文件块的数量。可对每个用户或组分别建立文件（i 节点）数量和 1 KB 块数量的限制。将建立一个软限制（首选）和一个硬限制。

例如，用户 **joe_doe** 在包含他（或她）的 **HOME** 目录 **/tmp** 的根文件系统 (**/**) 上，可能具有 1000 个块和 200 个文件的软限制，以及 1200 个块和 300 个文件的硬限制。而在挂接的文件系统 **/mnt** 上，可能分别具有 100 和 120 个块的软硬限制，但没有明确的文件限制 (0)。

每个文件系统用建立时间限制来确定用户可超出软限制的时间。缺省的时间限制为一周（7 天）。

当某用户超出他（或她）的软限制时，会在该用户的终端发出警告。该用户可以超出软限制继续增加使用，直到他（或她）超出硬限制或建立的时间限制。一旦发生这些事件之一，将发送消息到该用户的终端，进一步的文件创建和（或）增加的块使用尝试将失败。此时，该用户必须将他（或她）对超出限制的使用降低至软限制以下，以恢复正常的操作。

超出配额限制的用户（通过 **login**）登录时，系统将提示其已超出配额，同时提示相应的补救措施。用户可使用 **quota** 命令（请参阅 *quota*(1)）随时查看当前的配额状态。

操作系统为每个已启用配额的文件系统维护配额限制和利用率的统计数据（请参阅 *mount*(1M) 和 *quotaon*(1M)）。

磁盘配额是通过 **edquota** 命令（请参阅 *edquota*(1M)）为每个文件系统和每个用户单独建立的。该命令还可用来建立允许用户超过其软限制的时间限制。缺省的时间限制为 1 周。

限制和利用率统计数据都静态地存储于文件 **quotas** 中，该文件位于限制和利用率统计数据有效的每个文件系统的根目录下。每当卸除受影响的文件系统或使用 **quotactl()** 系统调用（请参阅 *quotactl*(2)）时，该文件都将与内核信息进行同步。

通过向 **/etc/fstab** 中的选项列表添加 **quotas** 选项，可以在引导或挂接时自动启用配额（请参阅 *fstab*(4) 和 *mount*(1M)）。缺省情况下， **mount** 不会启用磁盘配额。

随后，可使用 **quotaoff** 和 **quotaon** 命令禁用和重新启用配额（请参阅 *quotaon*(1M)）。当禁用配额时，内核将不再维护利用率统计数据，而且 **quotas** 文件的利用率统计数据也将因文件系统活动而失效。禁用配额可提高性能，但在随后重新启用配额时，必须运行 **quotacheck** 命令（请参阅 *quotacheck*(1M)）以更新内核和 **quotas** 文件。

repquota 命令（请参阅 *repquota*(1M)）可显示当前的配额统计数据报告。有些相关性但独立的 **quot** 命令（请参阅 *quot*(1M)），可独立地收集并报告磁盘配额子系统的磁盘利用率。

mount 命令（请参阅 *mount*(1M)）可报告启用了配额的所有文件系统。

数据存储结构

dqblk 数据结构（在 **<quota.h>** 中定义），由 **quotactl()** 系统调用（请参阅 *quotactl*(2)）用来设置或获取配额信息。该结构包含了用于存储用户当前文件和块数以及针对特定文件系统的配额限制的字段。

struct dqblk 包含以下成员：

```
uint32_t  dqb_bhardlimit;    /* maximum # of disk blocks +1 */
uint32_t  dqb_bsoftlimit;    /* preferred limit on disk blocks */
uint32_t  dqb_curblocks;     /* current block count */
uint32_t  dqb_fhardlimit;    /* maximum # allocated files +1 */
uint32_t  dqb_fsoftlimit;    /* preferred file limit */
uint32_t  dqb_curfiles;      /* current # allocated files */
uint32_t  dqb_btimelimit;    /* time limit for excessive block use */
uint32_t  dqb_ftimelimit;    /* time limit for excessive files */
```

dqblk64 数据结构（在 **<quota.h>** 中定义），由 **quotactl()** 系统调用（请参阅 *quotactl*(2)）用来设置或获取 64 位文件系统的配额信息。该结构包含了用于存储用户当前文件和块数以及针对特定文件系统的配额限制的字段。请注意，VxFS 3.5 中 *dqb64_curblocks* 字段的跟踪功能最多只能达到 2 TB。

struct dqblk64 包含以下成员：

```
uint64_t  dqb64_bhardlimit;  /* maximum # of disk blocks +1 */
uint64_t  dqb64_bsoftlimit;  /* preferred limit on disk blocks */
uint64_t  dqb64_curblocks;   /* current block count */
uint64_t  dqb64_fhardlimit;  /* maximum # allocated files +1 */
uint64_t  dqb64_fsoftlimit;  /* preferred file limit */
uint64_t  dqb64_curfiles;    /* current # allocated files */
uint64_t  dqb64_btimelimit;  /* time limit for excessive block use */
uint64_t  dqb64_ftimelimit;  /* time limit for excessive files */
```

网络功能

在 NFS 文件系统上并不完全支持配额。但是，如果远程系统提供 RPC **rquotad** 服务（请参阅 *rquotad*(1M)），那么 **quota** 命令可报告磁盘配额有效的远程 NFS 文件系统的配额统计数据。

提供了 **rquotad** 以便能够与其他系统相互支持。

举例

初始设置

内核必须重新配置成支持磁盘配额；请参阅系统管理手册。可以执行磁盘配额的文件系统应该具有挂接选项 **rw** 和 **quota**，如 *mount*(1M) 和 *fstab*(4) 中所述。

对每个要启用配额的文件系统，执行以下任务：

1. 挂接文件系统。

2. 将 **quota** 添加至 **/etc/fstab** 中现有的选项列表。例如，将根 (/) 条目的字符串 **default** 改为 **default,quota**。这样，所有相关文件系统在系统重新引导时便会自动启用配额。

3. 在文件系统的挂接目录下创建 **quotas** 文件。例如，在 **/mnt** 文件系统下，运行命令

cpset /dev/null /mnt/quotas 600 root bin

4. 使用 **edquota** 命令（请参阅 *edquota*(1M)）建立一个或多个原型用户配额。

   如果想让系统中的一组用户具有相同的限制，可使用 **edquota** 为一个原型用户设置配额；然后用 **edquota -p** 命令将这些限制复制给这一组用户。

5. 使用 **quotaon** 打开文件系统的配额。例如，运行命令

   /usr/sbin/quotaon /mnt

6. 在文件系统中运行 **quotacheck** （请参阅 *quotacheck*(1M)）以记录当前的利用率统计数据。

添加新用户

向配额系统中添加一个新用户：

1. 使用 **edquota** 复制一个现有用户的配额。

2. 运行 **quotacheck** 。

向已建立的系统添加新的文件系统

对于新的文件系统，请重复“初始设置”下的第 1 步至第 5 步。

警告

HP-UX 在缺省情况下允许 **chown()** 。它可以干预磁盘配额机制。如果用户可以访问 **chown** 命令（请参阅 *chown*(1)）或 **chown()** 系统调用（请参阅 *chown*(2)），那么配额可能失效。 **setprivgrp** 命令（请参阅 *setprivgrp*(1M)）可用于限制对 **chown()** 系统调用的访问，以便仅允许一组指定的用户使用 **chown** 命令或 **chown()** 系统呼叫。

sam 命令（请参阅 *sam*(1M)）还不支持磁盘配额。在添加新用户或文件系统时，任何所需的配额都必须在 **sam** 之外建立。

HP 在原始实现中添加了用于确保在由 **mount** 启用配额和由 **umount** 禁用配额（请参阅 *mount*(1M)）时配额文件内容正确性的功能，因此可以不必运行 **quotacheck** （请参阅 *quotacheck*(1M)）。然而，如果使用 **quotaoff** 和 **quotaon** （请参阅 *quotaon*(1M)）来控制配额，那么这些功能都将无效。

quotacheck 只能用于静止的文件系统以确保获得精确的使用信息。 **fsclean** 命令（请参阅 *fsclean*(1M)）的 **-qv** 选项可报告配额信息当前的生存能力。

作者

磁盘配额由 HP、加州大学伯克利分校和 Sun Microsystems 联合开发。

文件

/etc/fstab	有关文件系统的静态信息
/etc/mnttab	挂接的文件系统表
directory/**quotas**	

directory/**quota.group** 各个文件系统的用户和组配额统计数据静态存储位置，其中 *directory* 是文件系统的根目录，如指定给 **mount** 命令（请参阅 *mount*(1M)）。

另请参阅

chown(1)、 quota(1)、 edquota(1M)、 mount(1M)、 quot(1M)、 quotacheck(1M)、 quotaon(1M)、 rquotad(1M)、 setprivgrp(1M)、 chown(2)、 quotactl(2)、 vfsmount(2)、 fstab(4)。

名称

rbac: RBAC - 基于角色的访问控制

说明

RBAC（基于角色的访问控制）是基于超级用户的传统系统的全有或全无安全模型的替代方式。使用 RBAC，管理员可以将角色分配给非超级用户或 UNIX 组。每个角色的授权由操作和对象组成，其中操作是可以对对象执行的操作，对象是用户可以使用给定操作进行访问的对象。 HP-UX RBAC 数据库文件安装在 **/etc/rbac** 目录中。

下面是 HP-UX RBAC 命令列表，以它们的通常使用顺序列出：

roleadm 在 **roles** 、 **user_role** 和 **role_auth** 数据库中创建并管理与角色相关的信息。

authadm 在 **auths** 、 **role_auth** 和 **cmd_priv** 数据库中创建并管理授权信息。

cmdprivadm 在 **cmd_priv** 数据库中创建并管理命令的授权和权限信息。

rbacdbchk 验证语法以及所有 HP-UX RBAC 数据库之间的交叉引用，并在所有 RBAC 数据库之间执行交叉引用检查。

privrun 针对具有适当授权的用户执行权限命令。

privedit 允许具有适当授权的用户调用编辑器编辑受限文件。

下面是配置角色和授权的主要步骤：

1. 使用 **roleadm** 命令创建角色。角色将添加到 **/etc/rbac/roles** 数据库中。
2. 使用 **authadm** 命令添加授权。授权将添加到 **/etc/rbac/auths** 数据库中。
3. 使用 **authadm** 命令将授权或子角色分配给角色。角色、子角色和授权将添加到 **/etc/rbac/role_auth** 数据库中。
4. 使用 **roleadm** 命令将用户或 UNIX 组与角色关联。用户或组及其相应角色将添加到 **/etc/rbac/user_role** 数据库中。
5. 定义将要使用 **cmdprivadm** 命令与授权相关联的命令或文件以进行编辑。这些命令将添加到 **/etc/rbac/cmd_priv** 数据库中。
6. 使用 **rbacdbchk** 命令检查数据库。
7. 然后，授权的用户可以使用 **privrun** 包装命令运行权限命令，或使用 **privedit** 包装命令编辑受限文件。

privrun 包装命令确定给定命令所需的授权。这些授权－命令信息在 **/etc/rbac/cmd_priv** 数据库文件中定义。**privrun** 会查询 **roles** 和 **auths** 数据库文件，以根据通过 UNIX 组直接或间接分配给用户的角色，确定调用 privrun 的用户是否拥有必需的授权。

有关 **privrun** 命令的详细信息，请参阅 *privrun*(1M) ；有关 **cmd_priv** 数据库文件的信息，请参考 *privrun*(1M) 中的 **/etc/rbac/cmd_priv** 一节。

在确定编辑给定文件所需的授权方面， **privedit** 包装命令与 **privrun** 命令的工作方式类似。此授权－文件信息在

/etc/rbac/cmd_priv 数据库文件中定义。 **privedit** 会查询 **roles** 和 **auths** 数据库文件，并根据通过 UNIX 组直接或间接分配给用户的角色，确定调用 privedit 的用户是否拥有编辑文件所必需的授权。

有关 **privedit** 命令的详细信息，请参阅 *privedit*(1M) ；有关 **cmd_priv** 数据库文件的信息，请参考 *privedit*(1M) 中的 **/etc/rbac/cmd_priv** 一节。

数据库

在每个 HP-UX RBAC 数据库中，条目中的空白符将被忽略（这不包括用作记录分隔符的换行符 (\n)）。

HP-UX RBAC 数据库中的所有字段均区分大小写。

下面是当前提供的 HP-UX RBAC 数据库列表:

- **/etc/rbac/cmd_priv**
- **/etc/rbac/roles**
- **/etc/rbac/auths**
- **/etc/rbac/user_role**
- **/etc/rbac/role_auth**

有两个 HP-UX RBAC 数据库文件定义有效角色和授权。 **/etc/rbac/roles** 数据库定义有效角色， **/etc/rbac/auths** 数据库定义有效授权。授权按 (*operation*, *object*) 对的格式指定。

其他两个数据库文件将角色分配给用户或 UNIX 组，并将授权分配给角色。 **/etc/rbac/user_role** 将用户或 UNIX 组映射到其分配的角色。 **/etc/rbac/role_auth** 为每个角色定义授权集或子角色集。

/etc/rbac/cmd_priv 数据库将命令或文件与授权关联。

/etc/rbac/cmd_priv 数据库将命令或文件与授权关联。

/etc/rbac/cmd_priv

/etc/rbac/cmd_priv 文件包含执行某些命令或编辑某些文件所需的授权。它还具有与命令相关联的生成的权限（实际和有效 UID 和 GID、精细划分的权限和区块）。如果在成功授权之前需要用户进行重新验证，则在此文件中指定一个 PAM 服务名称，以指明 **privrun** 或 **privedit** 应如何向 PAM 标识其自身。

该文件包含任意数量的条目，其中每个条目在单行中按下列格式指定:

command | *file*:*arguments* :(*operation*, *object*) : *ruid*/*euid*/*rgid*/*egid* : *compartment* :*privs* : *pam service :" flags"*

privrun(1M) 和 *privedit*(1M) 中对这些字段进行了说明。

可能有多个条目使用同一命令行（所需的授权和生成的权限不同）。 **privrun** 和 **privedit** 按照文件中指定的顺序对每个条目进行评估，仅当用户没有所需的授权时，它才继续评估下一条目。

如果用户需要特定的条目，可以使用带有 **privrun** 或 **privedit** 的命令行选项来为特定条目指定权限集或授权集。请注意，对于 privedit，只可以指定授权。

有关 **/etc/rbac/cmd_priv** 数据库的详细信息，请参阅 *privrun*(1M) 或 *privedit*(1M) 。

/etc/rbac/roles

/etc/rbac/roles 数据库包含系统中所有有效角色的定义。在新角色可以分配给用户之前，管理员必须先在此文件中对其进行定义。

授权的用户可以使用 **roleadm** 命令，在 **/etc/rbac/roles** 文件中添加和删除角色（请参阅 *roleadm*(1M)）。

/etc/rbac/roles 数据库包含任意数量的条目，其中每个条目在单行上按下列格式定义:

rolename[**:***comment*]

按以下方式定义这些字段:

字段	说明
rolename	角色名称。例如，administrator、accountant、engineer、manager 等等。
[**:***comment*]	（可选）可选的简单注释或指向角色详细说明的可选 uri。

例如:

administrator:uri=http://www.site.com/adm.html

下例仅有角色名称，而没有注释或可选 uri:

SecurityOfficer

/etc/rbac/auths

/etc/rbac/auths 数据库包含系统中所有有效授权的定义，其格式为 (*operation*, *object*) 对。仅当管理员在此文件中定义了新的 (*operation*, *object*) 对后，才可以将 (*operation*, *object*) 对分配给角色。

授限的用户可以使用 **authadm** 命令在 **/etc/rbac/auths** 文件中添加和删除授权（请参阅 *authadm*(1M)）。

/etc/rbac/auths 数据库包含任意数量的条目，其中每个条目在单行上按下列格式定义:

(*operation***,** *object*)[**:***comment*]

按以下方式定义这些字段:

字段	说明
operation	表示可以对对象执行的操作。例如，**hpux.printer.add** 是添加打印机的操作。**hpux.printer.delete** 是删除打印机的操作。
object	用户可以使用给定操作进行访问的对象。如果指定 *，则操作可以访问所有对象。
[**:***comment*]	（可选）可选的简单注释或指向角色详细说明的可选 uri。

例如:

(hpux.printer.add, bldg7printer):Add printers in building 7.
(hpux.printer.delete, *):uri=http://foo.bar.com/printerauths.htm

(hpux.fs.backup, /dev/rdsk/c0t1d0):Backup physical disk 1

注释： **/etc/rbac/auths** 文件中指定的操作必须是完全限定的，且不可以使用通配符；但是，可以使用星号 **(*)** 通配符指定对象。使用匹配操作验证包含通配符操作的授权。要将授权分配给角色，必须至少有一个操作与通配符匹配。

/etc/rbac/user_role

/etc/rbac/user_role 数据库定义每个指定用户或 UNIX 组允许的角色。

授权的用户可以使用 **roleadm** 命令在 **/etc/rbac/user_role** 文件中添加和删除从用户到角色的映射定义（请参阅 *roleadm*(1M)）。

/etc/rbac/user_role 数据库包含任意数目的条目，其中每个条目在单行上按下列格式定义：

user-name | **&***group-name***:** *role*[**,***role*...]

按以下方式使用这些字段：

字段	说明
user-name | **&***group-name*	有效的用户名或 UNIX 组名。组名必须以 & 字符 **(&)** 开头。
role	**/etc/rbac/roles** 中定义的有效角色名。对于用户或组，可以指定多个角色，但条件是以逗号分隔这些角色。

下例显示用户 **Michael** 具有 administrator 和 programmer 的角色。同时，它还显示用户 **Jenny** 分配有 **SecurityOfficer** 角色。最后，它显示 UNIX 组 **users** 分配有 **RegularUser** 角色：

```
# roleadm list
Michael:Administrator, Programmer
Jenny:SecurityOfficer
&users:RegularUser
```

/etc/rbac/role_auth

/etc/rbac/role_auth 文件定义每个指定角色的授权和（或）子角色。每个授权按 (*operation*, *object*) 对的格式指定。授权对在 **/etc/rbac/auths** 数据库文件中定义。

子角色只是其他具有授权的角色。将子角色分配给角色时，该角色将继承此子角色的所有授权。子角色名称必须在 **/etc/rbac/roles** 数据库文件中定义。不允许递归角色定义。例如，如果“role1”具有子角色“role2”，并且用户通过 **roleassign** 将“role1”分配到“role2”，则这将会导致对“role1”和“role2”进行递归定义，从而 **roleassign** 命令将会失败。

授权的用户可以使用 **authadm** 命令在 **/etc/rbac/role_auth** 中指定每个角色的授权和（或）子角色（有关详细信息，请参阅 *authadm*(1M)）。

与角色相关联的所有授权和（或）子角色都必须在单个条目中指定。此条目可以有多行；但是，每个授权对不得超过一行。以字母数字字符后接冒号 (:) 开头的行被视为新条目。条目采用以下格式：

role: (*operation*, *object*) *subrole...*

role: (*operation*, *object*)...

role: subrole (*operation*, *object*)...

role: subrole subrole...

按以下方式定义这些字段:

字段	说明
role	**/etc/rbac/roles** 中定义的有效角色。
operation	可以对对象执行的特定操作。例如， **hpux.printer.add** 是添加打印机的操作。或者， **hpux.printer.*** 是添加或删除打印机的操作。
object	用户可以访问的对象。如果指定 *，则操作可以访问所有对象。 可为一个角色指定多个 (*operation*, *object*) 对。
subrole	**/etc/rbac/roles** 中定义的有效角色。该角色将被分配给其他角色。

以下行说明角色 **SecurityOfficer** 具有授权 (**hpux.passwd**, **/etc/passwd**)，这表示操作 **hpux.passwd** 可以访问对象 **/etc/passwd** 。 **SecurityOfficer** 还具有添加和删除系统用户的功能。

```
SecurityOfficer:   (hpux.passwd, /etc/passwd)
                   (hpux.user.add, *)
                   (hpux.user.del, *)
```

PrinterAdm 具有对所有对象执行 **hpux.printer.add** 的授权。

```
PrinterAdm:        (hpux.printer.add, *)
```

管理员拥有子角色 SecurityOfficer 和 PrinterAdm，因此，也就拥有前面的示例中显示的这两个子角色的所有授权。

```
Administrator:     SecurityOfficer PrinterAdm
```

/etc/rbac/aud_filter

/etc/rbac/aud_filter 文件定义角色和授权审核过滤。对于在此文件中找到其角色和关联授权的用户，将生成审核记录。如果在该文件中找不到某个用户的角色和关联授权，则不会生成特定于角色和授权的审核记录。每个授权按 operation, object 对的格式指定。

授权的用户（ **/etc/rbac/cmd_priv** 数据库文件中指定的用户）可以编辑 **/etc/rbac/aud_filter** ，以指定要审核的角色和授权。

所有与角色关联的授权都必须在单个条目中指定。每个角色只可以指定一个授权。条目采用以下格式:

*role***,** *operation***,** *object*

按以下方式定义这些字段:

字段	说明
role	**/etc/rbac/roles** 中定义的有效角色。
operation	可以对对象执行的特定操作。例如, **hpux.printer.add** 是添加打印机的操作。或者, **hpux.printer.*** 是添加或删除打印机的操作。
object	用户可以访问的对象。如果指定 * ,则操作可以访问所有对象。

下行指定审核具有授权 (**hpux.passwd**, **/etc/passwd**) 的角色 **SecurityOfficer** 。同时,还指定审核具有可以对所有对象执行 **hpux.printer.add** 的授权的 **PrinterAdm** 角色。

SecurityOfficer, hpux.passwd, /etc/passwd
PrinterAdm, hpux.printer.add, *

举例

下例显示超级用户如何使用 RBAC 管理命令来允许非超级用户 John 执行 **/usr/sbin/useradd** 命令。

1. 将名为 UserAdmin 的角色添加到角色数据库中:

   # /usr/sbin/roleadm add UserAdmin

   以上命令将 UserAdmin 角色添加到 **/etc/rbac/roles** 数据库中。

2. 列出系统中定义的授权,以确定哪些授权可用。

   # /usr/sbin/authadm list sys

3. 将名为 **(hpux.admin.useradd,*)** 的授权添加到 auths 数据库中。操作是 **hpux.admin.useradd** ,对象是 *。

   # /usr/sbin/authadm add hpux.admin.useradd

   上例中没有指定对象,因此对象将缺省为 * ,这表示操作可应用于所有对象。 **(hpux,admin.useradd,*)** 将添加到 **/etc/rbac/auths** 数据库中。

4. 将授权 **(hpux.admin.useradd,*)** 分配给 **UserAdmin** 角色。

   # /usr/sbin/authadm assign UserAdmin hpux.admin.useradd

   以上命令将以下条目添加到 **/etc/rbac/role_auth** 数据库中: **UserAdmin:(hpux.admin.useradd,*)**

5. 将角色 **UserAdmin** 分配给用户 **John** :

   # /usr/sbin/roleadm assign John UserAdmin

   以上命令将以下条目添加到 **/etc/rbac/user_role** 数据库中: "John:UserAdmin"

6. 将命令 **/usr/sbin/useradd** 添加到 **cmd_priv** 数据库中：

   # /usr/sbin/cmdprivadm add cmd=/usr/sbin/useradd
   op=hpux.admin.useradd ruid=0 euid=0

   以上命令将以下条目添加到 **/etc/rbac/cmd_priv** 数据库中：

   /usr/sbin/useradd:dflt:(hpux.admin.useradd,*):0/0//:dflt:dflt:dflt:

7. 检查 RBAC 数据库中的语法和条目是否有效：

   # /usr/sbin/rbacdbchk

8. 现在，非超级用户 John 已与 UserAdmin 角色相关联。已经为 UserAdmin 角色分配了名为 **(hpux.admin.useradd, *)** 的授权，该授权是按照 **cmd_priv** 数据库中添加的每个条目执行 **/usr/sbin/useradd** 时所必需的授权。现在，非超级用户 John 可以使用 **privrun** 运行 **/usr/sbin/useradd** ，以便按以下方式将常规用户添加到系统中：

   # /usr/bin/privrun /usr/bin/useradd new_user

审核

命令 *privrun*(1M) 、 *roleadm*(1M) 、 *authadm*(1M) 和 *cmdprivadm(1M)* 都可以生成审核记录。审核记录包括调用方的用户名、 UID、角色、授权、对象、事件时间、事件的成功或失败等等。

可以提供审核过滤数据库文件 (**/etc/rbac/aud_filter**) ，用户可以使用该文件指定要审核的角色和授权 (*operation*, *object*) 。仅当调用方的角色和授权与 **/etc/rbac/aud_filter** 数据库中的其中一个条目匹配时，才生成角色-授权审核记录。

如果审核过滤数据库文件不存在或无法访问，则仍会生成审核记录。但是，如果审核过滤数据库文件存在但为空，则不生成审核记录。

下面是有关如何为 **roleadm**

生成和显示审核记录的示例：

audevent -Pfe admin
audsys -f
audsys -n -c /tmp/aud.out -s 2048
roleadm add new_role_1
audsys -f
audisp /tmp/aud.out

请参阅 *audit*(5) 、 *audevent*(1M) 、 *audsys*(1M) 和 *audisp*(1M) ，以了解有关生成和显示审核记录的详细信息。

文件

/etc/rbac/auths 包含所有有效授权定义的数据库。

/etc/rbac/cmd_priv 一个数据库，包含执行指定命令或编辑特定文件所需的授权，以及为命令执行而更改 UID 和 GID 的权限。

/etc/rbac/roles 包含所有角色的所有有效定义的数据库。

/etc/rbac/role_auth 为每个角色定义授权和（或）子角色的数据库。

/etc/rbac/user_role 为每个特定用户或 UNIX 组指定角色的数据库。

/etc/rbac/aud_filter 包含要审核的角色和关联授权的列表的数据库。

另请参阅

authadm(1M)、 cmdprivadm(1M)、 privrun(1M)、 privedit(1M)、 rbacdbchk(1M)、 roleadm(1M)、 privileges(5)、compartments(5)。

名称

rcsintro - RCS 命令的说明

说明

版本控制系统 (RCS) 可使对 ASCII 文本文件修订版的存储、检索、日志记录、标识和合并等操作自动化。 RCS 对于管理需要频繁修订的文件非常有用。

RCS 的功能

- 文本文件修订版的存储和检索。 RCS 以一种节省空间的方式存储修订版。可通过版本号、符号名称、日期、作者和状态来检索版本。
- 维护完整的历史变更记录。 RCS 可自动记录所有的更改。除了每个修订版的文本， RCS 还将存储作者、日期、签入时间，以及一条概述此次更改的日志消息。
- 解决访问冲突的方法。当两个或多个用户试图修改文件的同一个修订版时， RCS 将对他们发出警告以阻止一个修改损坏另一个修改。
- 维护修订版树。 RCS 可为每个文件维护单独的发展线路。它以树结构存储，从而可以表示各版本之间的发展关系。
- 修订版的合并以及冲突的解决方法。一个文件的两个单独发展线路可以通过合并操作合而为一。如果待合并的修订版影响更改了文件中的相同行， RCS 将标记出这些重叠的更改。
- 发布和配置控制。可为修订版指定符号名称，并将其标记为已发布、稳定、实验等等。这样，就可以简单直接地描述文件的配置。
- 使用文件名、版本号、创建时间、作者等自动标识各个修订版。这个标识就像邮戳一样可以嵌入修订版文本中合适的位置。这些邮戳可以简单地确定哪些文件的哪些修订版构成了给定的配置。
- 备用存储的最小化。 RCS 用很少的额外空间来存储修订版（仅存储差异）。如果删除了中间修订版，那么剩余的 delta 版也将相应地压缩。

RCS 入门

基本的用户接口非常简单。入门用户只需学习两条命令： **ci** 和 **co** （请参阅 *ci*(1) 和 *co*(1) ）。 **ci** 是 “**check in**（签入）”的缩写，它可将文本文件的内容存储到名为 RCS 的归档文件中。 RCS 文件包含了特定文本文件的所有修订版。 **co** 是 “**check out**（签出）”的缩写，它可从 RCS 文件中检索修订版。

假定希望将文件 **f.c** 置于 RCS 的控制中。调用签入命令：

ci f.c

这条命令将创建 RCS 文件 **f.c,v** ，将 **f.c** 作为修订版 **1.1** 存储到其中，并删除 **f.c** 。它还要求给出说明。说明应是该文件的内容概要。所有后续的签入命令都要求给出一个日志条目，以汇总所做的更改。

名称以“ **,v** ”结尾的文件称为 RCS 文件（ **v** 代表“版本”）；所有其他文件假定为工作文件。要恢复上例中的工作文件 **f.c** ，可使用签出命令：

co f.c

该命令将从 **f.c,v** 中提取最新修订版并将其写入 **f.c** 中。此时可编辑 **f.c** ，也可以通过调用以下命令将其重新签入：

ci f.c

ci 将适当地增加版本号。如果 **ci** 发出以下错误信息：

ci error: no lock set by *your-login*

系统管理员决定在锁定属性设置为“严格”时创建所有 RCS 文件。这种情况下，在上一次签出中，应该已经锁定了修订版。最后一次签出应该是：

co -l f.c

当然，现在签出为锁定状态已经太晚了，因为可能已经修改了 **f.c** ，而第二次签出将覆盖修改。而应调用：

rcs -l f.c

该命令将帮助锁定最新版本，除非其他人已经将其锁定。在这种情况下，只能去和此人协商。

锁定可确保您且只有您可以签入下一次更新，从而解决了多人修改同一个文件的苦恼问题。即使一个修订版被锁定，仍可签出它进行读取、编译等操作。锁定只是阻止除锁定者外的其他人签入。

如果您的 RCS 文件是专用的，也就是说，只有您将修订版存入其中，那么就不需要严格锁定，您可将其关闭。如果关闭了严格锁定，那么 RCS 文件的所有者在签入时不需要锁；但其他用户仍然需要锁。使用以下命令来开启和关闭严格锁定：

rcs -U f.c

和

rcs -L f.c

如果不想让 RCS 文件使工作目录混乱，可以在工作目录下创建一个名为 **RCS** 的子目录，然后将所有的 RCS 文件移至那里。 RCS 命令将搜索那个目录以查找所需的文件。上述所有的命令都可以不加修改的情况下使用。

为了避免在签入过程中删除工作文件（当希望继续编辑时），请调用：

ci -l f.c

或者

ci -u f.c

这些命令按常规签入了 **f.c** ，但执行了隐式的签出。第一种形式同样锁定了签入的修订版，但第二种形式没有锁定。如此，这些选项帮您节省了一步签出操作。第一种形式在严格锁定的情况下有用；第二种形式在非严格锁定的情况下有用。这两种形式都会更新工作文件中的标识标记（请参阅下文）。

可以为 **ci** 指定想要指定给签入的修订版的编号。假定所有版本号是 1.1，1.2，1.3 等等，但您希望从版本 2 开始。命令：

ci -r2 f.c

或者

ci -r2.1 f.c

将为新的修订版指定编号 2.1。从那时起， **ci** 将以 2.2，2.3 等作为后续修订版的编号。相应的 **co** 命令：

co -r2 f.c

和

co -r2.1 f.c

分别检索编号为 2.x 的最新修订版和 2.1 修订版。不带版本号的 **co** 从“树干”上选择最新版本；也就是说，由两个字段组成的版本号最高的修订版。多于 2 个字段的版本号需要分支。例如，要在修订版 1.3 处开始一个分支，请调用：

ci -r1.3.1 f.c

这个命令在修订版 1.3 处开始了一个编号为 1 的分支，并将编号 1.3.1.1 指定给新修订版。有关分支的详细信息，请参阅 *rcsfile*(4) 。

RCS 文件命名和位置

RCS 可识别两种文件： RCS 文件（修订版归档）和工作文件。工作文件名由 RCS 的用户定义， RCS 文件名由 RCS 将“ **,v** ”附加到工作文件名的末尾生成。可以 3 种方式指定 RCS 文件和工作文件对：

- 同时给出 RCS 文件和工作文件。 RCS 文件名的格式是 *path1***/***workfile***,v** ，而工作文件名的格式是 *path2***/***workfile* ，其中 *path1* 和 *path2* 都是（可能不同也可能为空）路径，而 *workfile* 是文件名。
- 仅给出 RCS 文件。此时假定工作文件就在当前目录下，且它的名字由 RCS 文件名去掉 *path1***/** 和后缀“ **,v** ”派生而来。
- 仅给出工作文件。此时 RCS 的文件名由工作文件名去掉 *path2***/** 并附加后缀“ **,v** ”派生而来。

如果 RCS 文件名被省略了或者并未指定路径，那么 RCS 命令将在目录 **./RCS** （或者如果它是目录链接，则为它指向的目录）中查找 RCS 文件，如果没有，再到当前的工作目录中查找。

RCS 目录链接

RCS 支持目录链接。如果当前的工作目录中存在一个名为 RCS 的常规文件，那么 RCS 会将文件的第一行解释为存储 RCS 文件的目录路径名。 RCS 可以沿着一条多达 10 个目录链接的链到达 RCS 目录。

自动标识

RCS 可将用于标识的特殊字符串放入源代码和对象代码中。要获得这些标识，请放置标记：

$Header$

到您的文本中，例如放在注释中。 RCS 将这个标记替换为如下形式的字符串：

$Header: *filename revision_number date time author state* **$**

如果每个模块的第一页都有这样的标记，那么您就始终可以看到自己操作的是哪个修订版。 RCS 可自动更新这些

标记。要将这些标记扩展到对象代码中，只需将它们转换成文字字符串。在 C 中，可使用如下操作:

static char rcsid[] = "$Header$";

命令 **ident** 可从任意文件中提取这些标记，即使是对象代码和转储。如此， **ident** 可以让您了解给定的程序使用的是哪些模块的哪些修订版。

您还会发现将标记 **Log** 放入您的注释文本中也非常有用。这个标记将累积签入时请求的日志消息。这样，用户可直接在其中维护其文件的完整历史。还有几种其他的标识标记。有关详细信息，请参阅 *co*(1) 。

警告

RCS 文件的名称是通过将 **,v** 附加到工作文件名的末尾而生成的。如果得到的 RCS 文件名对于 RCS 文件应驻留的文件系统而言太长，则 RCS 命令将终止并生成错误消息。

RCS 设计为仅与 TEXT 文件一起使用。如果试图将 RCS 与非文本（二进制）文件一起使用，则会导致数据损坏。

作者

rcsintro 由 Purdue University, West Lafayette, IN 47907 的 Walter F. Tichy 开发。

另请参阅

ci(1)、 co(1)、 ident(1)、 merge(1)、 rcs(1)、 rcsdiff(1)、 rcsmerge(1)、 rlog(1)、 rcsfile(4)。

“Design, Implementation, and Evaluation of a Revision Control System”，作者：Walter F. Tichy，发表于《Proceedings of the 6th International Conference on Software Engineering》，IEEE，东京，1982 年 9 月。

名称

regexp - 正则表达式和模式匹配表示法定义

说明

“正则表达式”是很多实用程序支持的机制，用于在文本中定位和操作模式。 “模式匹配表示法”由 Shell 和其他实用程序用于文件名扩展。此手册条目定义了两种形式的正则表达式： “基本正则表达式”和 “扩展正则表达式”以及一种形式的 “模式匹配表示法”。

基本正则表达式

基本正则表达式 (RE) 表示法和构建规则适用于定义为使用基本 RE 的实用程序。下列规则的所有例外都在使用 RE 的特定实用程序的描述中进行了注释。

匹配单个字符的 RE

下列 RE 匹配单个字符或单个排序元素：

普通字符

普通字符是匹配其自身的 RE。普通字符是支持的字符集中除了换行符和在下面的 特殊字符中列出的正则表达式特殊字符以外的任何字符。前面有反斜杠 (\ ) 的普通字符被视作普通字符自身，除非该字符是 **(** 、 **)** 、 **{** 或 **}** ，或者是数字 **1** 到 **9** （请参阅 匹配多个字符的 **RE** ）。匹配是基于用于对字符进行编码的位模式；而不是该字符的图形形式。

特殊字符

前面有反斜杠的正则表达式特殊字符是匹配该特殊字符自身的正则表达式。前面没有反斜杠时，这样的字符在 RE 规范中有特殊含义。正则表达式特殊字符以及在其中有特殊含义的环境有：

. [**　句点、左方括号和反斜杠除了用在括号表达式中以外都是特殊字符（请参阅 **RE 括号表达式）。

*****　星号除了在以下情况中以外是特殊字符：用在括号表达式中、作为正则表达式的第一个字符，或者作为字符对 **\\(** 后面的第一个字符（请参阅 匹配多个字符的 **RE** ）。

^　插字符号在用作整个 RE 的第一个字符（请参阅 表达式定位）或者作为括号表达式的第一个字符时是特殊字符。

$　美元符号在用作整个 RE 的最后一个字符时是特殊字符（请参阅 表达式定位）。

delimiter　用于限定（即分隔）整个 RE 的所有字符对该 RE 来说都是特殊字符。

句点

句点 (.)，当用在括号表达式以外时，是匹配除了换行符以外的所有可输出或不可输出字符的 RE。

RE 括号表达式

方括号 ([ ]) 中的括号表达式是一个 RE，它匹配由括号表达式所表示的排序元素非空集中包含的单个排序元素。

下列规则适用于括号表达式：

“括号表达式” 括号表达式是 “匹配列表表达式” 或 “非匹配列表表达式”，并且以任意顺序包含一个或多个表达式。表达式可以是：排序元素、排序符号、非排序字符、等价类、范围表达式或字符类。右方括号 ( ]) 如果在列表中第一个出现（如果有初始 ^ ，则位于其后），则失去它的特殊含义并在括号表达式中代表它自身。否则，它将终止该括号表达式（除非它是有效排序符号、等价类或字符类的结尾右方括号，或者它是排序符号或等价类表达式中的排序元素）。特殊字符

**. * [**

（句点、星号、左方括号和反斜杠）在括号表达式中失去它们的特殊含义。

字符序列：

[. [= [:

（左方括号后面跟句点、等号或冒号）是括号表达式中的特殊字符，用于分隔排序符号、等价类表达式和字符类表达式。这些符号后面必须跟有有效的表达式和匹配的终止 **.]** 、 **=]** 或 **:]** 。

“匹配列表” 匹配列表表达式指定了一个列表，该列表匹配其中列出的任意一个字符。该列表中的第一个字符不能是插字符号。例如， **[abc]** 是匹配 **a** 、 **b** 或 **c** 中任意一个的 RE。

“非匹配列表 “非匹配列表”表达式以插字符号 (^) 开头，并指定一个列表，该列表匹配 *except* 换行符和该列表中列出字符以外的任意字符或排序元素。例如， **[^abc]** 是匹配除了换行符或者 **a**、 **b** 或 **c** 以外的 RE。插字符号 只有当紧跟在左方括号后面在列表中第一个出现时，才有此特殊含义。

“排序元素” “排序元素”是一个或多个字符的序列，该序列代表排序序列中的单个元素，而排序序列通过环境变量 **LC_COLLATE** 的最新设置来标识（请参阅 *setlocale*(3C) ）。

“排序符号” “排序符号”是包含在方括号句点 （ [. .] ） 分隔符中的排序元素。多字符排序元素必须以排序符号表示，以便将它们与单个字符排序元素相区分。例如，如果字符串 **ch** 是有效的排序元素，那么 **[[.ch.]]** 被视作匹配相同字符串的元素，而 **ch** 被视作字符 **c** 和 **h** 的简单列表。如果方括号句点分隔符中的字符串不是当前排序序列定义中的有效排序元素，那么该符号被视作无效表达式。

“非排序字符” “非排序字符”是排序要忽略的字符。根据定义，这样的字符不能出现在等价类或范围表达式中。

“等价类” “等价类”表达式代表属于一个等价类的排序元素集。它通过用方括号等号 ([= =]) 分隔符包含等价类中的任意一个排序元素来表示。例如，如果 **a** 和 **A** 属于同一等价类，那么 **[[=a=]b]** 和 **[[=A=]b]** 等价于 **[aAb]** 。

“范围表达式” “范围表达式”代表位于当前排序序列中两个元素之间的排序元素集，而当前排序序列通过环境变量 **LC_COLLATE** 的最新设置定义（请参阅 *setlocale*(3C) ）。它通过以连字符 (-) 分隔的起始点和结束点表示。

起始范围点和结束范围点必须是排序元素、排序符号或等价类表达式。用作范围表达式结束点的 *equivalence class expression* ，被解释为该等价类中的所有排序元素都被包括在范围中。例如，如果排序顺序是 **A** 、 **a** 、 **B** 、 **b** 、 **C** 、 **c** 、 **ch** 、 **D** 和 **d** ，并且字符 **A** 和 **a** 属于同一等价类，那么表达式 **[[=a=]-D]** 将被视作 **[AaBbCc[.ch.]D]** 。

起始和结束范围点都必须是有效的排序元素、排序符号或等价类表达式，并且结束范围点必须排序等于或高于起始范围点；否则该表达式无效。例如，针对上面的排序顺序同时假设 **E** 是非排序字符，则表达式 **[[=A=]-E]** 和 **[d-a]** 都无效。

结束范围点也可以是后续范围表达式中的起始范围点。每个这样的范围表达式分别进行计算。例如，括号表达式 **[a-m-o]** 被视作 **[a-mm-o]** 。

连字符如果在列表中第一个出现（如果有初始 ^ ，则位于其后）或最后一个出现，或者作为范围表达式中最右边的符号，则它代表连字符自身。例如，表达式 **[-ac]** 和 **[ac-]** 等价并且匹配字符 **a** 、 **c** 或 **-** 中的任意字符；表达式 **[^-ac]** 和 **[^ac-]** 等价并且匹配除了换行符、 **a** 、 **c** 或 **-** 以外的任意字符；表达式 **[%--]** 匹配已定义排序序列中 **%** 和 **-** 之间（含边界值）的任意字符；表达式 **[--@]** 匹配已定义排序范围中 **-** 和 **@** 之间（含边界值）的任意字符；而表达式 **[a--@]** 无效，如果在排序序列中 **-** 位于 **a** 之前的话。

如果括号表达式必须同时指定 **-** 和 **]** ，则 **]** 必须放在最前面（如果存在 ^ ，则位于其后），而 **-** 放在括号表达式的最后。

“字符类” 字符类表达式表示属于一个字符类的字符集，该字符类通过环境变量 **LC_CTYPE**. 的最近设置定义。它通过包含在方括号冒号 ([: :]) 分隔符中的字符类名称表示。

所有环境中都支持的标准字符类表达式有:

[:alpha:]	字母
[:upper:]	大写字母
[:lower:]	小写字母
[:digit:]	十进制数字
[:xdigit:]	十六进制数字
[:alnum:]	字母或十进制数字
[:space:]	在显示文本中生成空格的字符
[:print:]	输出字符
[:punct:]	标点符号
[:graph:]	具有可见形式的字符
[:cntrl:]	控制字符

[:blank:] 空白字符

例如，如果环境变量 **LC_CTYPE** 设置为 **C** ，则表达式 **[[:upper:]]** 等价于 **[A-Z]** 。同样，表达式 **[[:digit:]]** 与 **[0-9]** 相同。

匹配多个字符的 RE

下列规则可以用于从匹配单个字符的 RE 构建匹配多个字符的 RE：

RE RE 连接多个 RE 构成了一个 RE，该 RE 匹配遇到的第一个由该 RE 每个组成部分所匹配的连接字符串。例如，RE **bc** 匹配字符串 **abcdefabcdef** 的第二个和第三个字符。

*RE** 后跟星号 (*) 的匹配单个字符的 RE 是匹配星号前 RE 零次或多次出现的 RE。将选择遇到的允许匹配的第一个字符串，并且该匹配字符串包含该 RE 允许的最大字符数。例如，在字符串 **abbbcdeabbbbbbcde** 中，RE **b*c** 和 RE **bbb*c** 都与第二到第五位置的子字符串 **bbbc** 相匹配。星号作为 RE 的第一个字符失去此特殊含义，并被视作字符本身。

\(*RE***\)** 子字符串可以通过在 RE 中括在字符对 **\(** 和 **\)** 之内来定义。这样的子表达式匹配在没有 **\(** 和 **\)** 的情况下将会匹配的任意字符。子表达式可以进行任意嵌套。紧跟在 **\(** 后面的星号失去它的特殊含义，并被视作字符自身。紧跟在 **\)** 后面的字符被视作无效字符。

*****n* 表达式 *****n* 匹配包含在位于 *****n* 之前 **\(** 和 **\)** 之间的子表达式匹配的相同字符串。字符 *n* 必须是从 **1** 到 **9** 的一个数字，指定第 *n*- 个子表达式（该子表达式以第 *n*- 个 **\(** 开始并以相应的成对 **\)** 结束）。例如，表达式 **^\(.*\)\1$** 匹配由两个相邻出现的相同字符串组成的行。

如果 *****n* 后面跟着一个星号，则它匹配所指子表达式的零次或多次出现。例如，表达式 **\(ab\(cd\)ef\)Z\2*Z\1** 匹配字符串 **abcdefZcdcdZabcdef** 。

*RE***\{***m***,***n***\}** 后面跟着 **\{***m***\}** 、 **\{***m***,\}** 或 **\{***m***,***n***\}** 的匹配单个字符的 RE 是匹配该 RE 重复出现的 RE。*m* 和 *n* 的值必须是范围在 0 到 255 之间的十进制整数，其中 *m* 指定出现的确切次数或最少次数，而 *n* 指定出现的最大次数。 **\{***m***\}** 匹配前面的 RE 正好出现 *m* 次， **\{***m***,\}** 匹配至少出现 *m* 次，而 **\{***m***,***n***\}** 匹配的出现次数为 *m* 和 *n* 之间（含边界值）的任意数。

将选定遇到的匹配该表达式的第一个字符串；它将包含尽可能多的 RE 出现次数。例如，在字符串 **abbbbbbbc** 中，RE **b\{3\}** 匹配第二到第四个字符，RE **b\{3,\}** 匹配第二到第八个字符，而 RE **b\{3,5\}c** 匹配第四到第九个字符。

表达式定位

RE 可以根据下列规则限定为匹配开始或结束一行的（即，定位的）字符串：

- 插字符号 (^) 作为一个 RE 的第一个字符将该表达式定位到一行的开始；只有从一行的起始字符开始的字符串可以匹配该 RE。例如，RE **^ab** 匹配行 **abcdef** 中的字符串 **ab** ，但是不匹配行 **cdefab** 中的相同字符串。
- 美元符号 ($) 作为一个 RE 的最后一个字符，将该表达式定位到一行的结尾；只有在行的最后一个字符结尾的字符串可以匹配该 RE。例如，RE **ab$** 匹配行 **cdefab** 中的字符串 **ab** ，但是不匹配行 **abcdef** 中的相同字符串。

- 同时被 ^ 和 **$** 定位的 RE 只匹配成行的字符串。例如，RE **^abcdef$** 只匹配由字符串 **abcdef** 组成的行。

在定位符号之后使用重复字符 (+,*) 是非法的。

扩展的正则表达式

扩展的正则表达式 (ERE) 表示法和构造规则适用于被定义为使用扩展 RE 的实用程序。下列规则的任何例外，都在使用 ERE 的特定实用程序的描述中进行了注释。

匹配单个字符的 **ERE**

下列 ERE 匹配单个字符或单个排序元素:

普通字符

普通字符是匹配其自身的 ERE。普通字符是支持的字符集中除了换行符和在下面的“特殊字符”中列出的正则表达式特殊字符以外的任何字符。前面有反斜杠 (\ ) 的普通字符被视作该普通字符自身。匹配是基于用于对字符进行编码的位模式，而不是该字符的图形形式。

特殊字符

前面有反斜杠的正则表达式特殊字符是匹配该特殊字符自身的正则表达式。前面没有反斜杠时，这样的字符在 ERE 规范中有特殊含义。扩展的正则表达式特殊字符以及在其中有特殊含义的环境有:

. [\ ( ) * + ? $ | 句点、左方括号、反斜杠、左圆括号、右圆括号、星号、加号、问号、美元符号以及竖线都是特殊符号，除非他们用于括号表达式中（请参阅 **ERE** 括号表达式）。

^ 除非用于括号表达式的非开始位置，否则，插字符号是特殊符号。

delimiter 用于限定（即分隔）整个 ERE 的所有字符对该 ERE 来说都是特殊字符。

句点

句点 (.)，当用在括号表达式以外时，是匹配除了换行符以外的所有可输出或不可输出字符的 ERE。

ERE 括号表达式

ERE 括号表达式的语法和规则与上面 RE 括号表达式的相同。

匹配多个字符的 **ERE**

下列规则可以用于从匹配单个字符的 ERE 构建匹配多个字符的 ERE:

ERE ERE 多个 ERE 的连接，匹配遇到的第一个匹配该 ERE 每个组成部分的字符串连接。这样的 ERE 连接包含在圆括号中时与没有圆括号时匹配相同的字符串。例如，ERE **bc** 和 ERE **(bc)** 都匹配字符串 **abcdefabcdef** 的第二和第三个字符。匹配最长的整体字符串。

*ERE***+** 特殊字符加号 (+)，当跟在匹配单个字符的 ERE 或圆括号内 ERE 连接的后面时，是一个 ERE，它匹配加号之前 ERE 的一次或多次出现。匹配的字符串会包含尽可能多的出现次数。例如，ERE **b+c** 匹配字符串 **acabbbcde** 中的第四到第七个字符。

*ERE*** 特殊字符星号 (*)，当跟在匹配单个字符的 ERE 或圆括号内 ERE 连接的后面时，是一个 ERE，它匹配星号之前 ERE 的零次或多次出现。例如，ERE **b*c** 匹配字符串 **cabbbcde** 中的第一个字符。如果有多个选择，则选定允许匹配的最长和最左边的字符串。例如，ERE

b*cd 匹配字符串 **cabbbcdebbbbbbcdbc** 中的第三到第七个字符。

*ERE***?** 特殊字符问号（?），当跟在匹配单个字符的 ERE 或圆括号内 ERE 连接的后面时，是一个 ERE，它匹配问号之前 ERE 的零次或多次出现。匹配的字符串会包含尽可能多的出现次数。例如，ERE **b?c** 匹配字符串 **acabbbcde** 中的第二个字符。

ERE{*m*,*n*} 与基本正则表达式语法 *ERE*\{*m*,*n*\} 功能相似的间隔表达式

替换

由特殊字符竖线 (|) 分隔的两个 ERE，匹配其中任何一个 ERE 匹配的字符串。例如，ERE **((ab)|c)d** 匹配字符串 **abd** 和字符串 **cd** 。竖线“|”不能出现在下列情况中：

不能出现在 ERE 的开头或结尾。

不能紧跟在一个竖线之后出现。

不能出现在左圆括号之后。

不能紧挨着出现在右圆括号之前。

优先级

优先级顺序如下，由高到低：

[ ]	方括号
*** + ?**	星号、加号、问号
^ $	定位
	连接
|	替换

例如，ERE **abba|cde** 被解释为“匹配 **abba** 或 **cde**”。它不意味着“匹配 **abb** 后面跟着 **a** 或 **c** 后面依次跟着 **de**”（因为连接具有比替换更高的优先级）。

表达式定位

ERE 可以根据下列规则限定为匹配开始或结束一行的（即，定位的）字符串：

- 插字符号（^）匹配行的开始（将表达式定位在行的开始）。例如，ERE **^ab** 匹配行 **abcdef** 中的字符串 **ab**，但是不匹配行 **cdefab** 中的相同字符串。
- 美元符号（$）匹配行的结尾（将该表达式定位在行的结尾）。例如，ERE **ab$** 匹配行 cdefab**,** 中的字符串 **ab**，但是不匹配行 **abcdef** 中的字符串。
- 同时被 ^ 和 **$** 定位的字符串只匹配成行的字符串。例如，ERE **^abcdef$** 只匹配由字符串 **abcdef** 组成的行。只有空行匹配 ERE **^$**。

在定位符号之后使用重复字符 (+,*) 是非法的。

模式匹配表示法

下列规则适用于除了在使用模式匹配的特定实用程序的描述中注释的模式匹配表示法之外的模式匹配表示法。

匹配单个字符的模式

下列模式匹配单个字符或单个排序元素:

普通字符

普通字符是匹配其自身的模式。普通字符是在支持的字符集中除了换行符和在下面的“特殊字符”中列出的模式匹配特殊字符以外的任何字符。匹配是基于用于对字符进行编码的位模式，而不是该字符的图形形式。

特殊字符

前面有反斜杠 (\ ) 的模式匹配特殊字符，是匹配该特殊字符自身的模式。前面没有反斜杠时，这样的字符在模式规范中有特殊含义。模式匹配特殊字符以及在其中有特殊含义的环境有:

? * [问号、星号和左方括号除了用在括号表达式以外，都是特殊字符（请参阅 模式括号表达式）。

问号

问号 (?) ，当用在括号表达式以外时，是匹配除了换行符以外的所有可输出或不可输出字符的模式。

模式括号表达式

除了下列例外，模式括号表达式的语法和规则与上面 RE 括号表达式的相同:

感叹号 (!) 取代了正则表达式表示法中非匹配列表中插字符号 (^) 的角色。

反斜杠用作括号表达式中的转义字符。

匹配多个字符的模式

下列规则可以用于从匹配单个字符的模式构建匹配多个字符的模式:

***** 星号 (*) 是匹配任意字符串，包括空字符串的模式。

RERE 匹配单个字符的模式的连接是匹配每个连接的模式所匹配的单个字符或排序元素的连接的有效模式。例如，模式 **a[bc]** 匹配字符串 **ab** 和 **ac** 。

一个或多个星号与一个或多个匹配单个字符的模式连接，是有效的模式。在这样的模式中，每个星号都匹配零个或多个字符的字符串，直到匹配该模式中星号之后字符的第一个字符。

例如，模式 **a*d** 匹配字符串 **ad** 、 **abd** 和 **abcd** ；但是不匹配字符串 **abc** 。当星号是模式中的第一个或最后一个字符时，它匹配由模式中的其他字符所匹配的字符串之前或之后的一个或多个字符。例如，模式 **a*d*** 匹配字符串 **ad** 、 **abcd** 、 **abcdef** 、 **aaaad** 和 **adddd** ；模式 ***a*d** 匹配字符串 **ad** 、 **abcd** 、 **efabcd** 、 **aaaad** 和 **adddd** 。

用于文件名扩展的模式规则限定条件

当模式匹配表示法由 *sh*(1) 、 *csh*(1) 、 *ksh*(1) 和 *make*(1) 用于文件名扩展时，上述模式匹配规则受下列规则限定。

如果文件名（包括跟在斜线 (/) 字符之后的路径名组成部分）以句点 (.) 开始，则必须用句点作为该模式的第一个字符来明确匹配句点；它不能用星号特殊字符、问号特殊字符或括号表达式匹配。此规则不适用于 *make*(1) 。

路径名中的斜线字符必须在该模式中使用斜线来明确匹配；它不能用星号特殊字符、问号特殊字符或括号表达式匹配。对于 *make*(1) ，只有跟在最后一个斜线字符之后的路径名部分可以用特殊字符匹配。也就是说，最后一个斜线字符之前的所有特殊字符都失去了它们的特殊含义。

相应地，指定的模式匹配现有文件名和路径名。如果该模式匹配任何现有的文件名或路径名，则该模式被替换为那些文件名和路径名，并根据发生作用的排序序列进行排序。如果该模式不匹配任何现有的文件名或路径名，则该模式字符串保持不变。

如果该模式以波浪符 (~) 开头，那么所有在第一个斜线之前的普通字符（或者所有字符，如果没有斜线）都被视作可能的登录名。如果登录名为空（即该模式只包含波浪符或者波浪符之后紧跟着斜线），那么该波浪符被替换为该进程主目录的路径名，后跟一个斜线。否则，波浪符和登录名的组合被替换为与该登录名相关的主目录的路径名，后跟一个斜线。如果系统无法标识登录名，则结果由实现定义。此规则不适用于 *sh*(1) 或 *make*(1) 。

如果该模式包含 **$** 字符，则可能发生变量替换。环境变量可在模式中嵌入为：

$*name*

或：

${*name***}**

大括号用于确保跟在 *name* 之后的字符不被解释为属于 *name* 。替换只按照指定的顺序发生一次；也就是说，对于由于替换而产生的新名称将不再检查结果字符串。

在 **case** 命令中使用的模式规则限定条件

当模式匹配表示法用于 case 命令 *sh*(1) 和 *ksh*(1) 中时，上述模式匹配规则由下列规则限定。

可以通过使用竖线字符 (|) 分隔单个模式在单个子句中指定多个替代模式；匹配任何以此方法分隔的模式的字符串将导致选中相应的命令列表。

另请参阅

ksh(1)、 sh(1)、 fnmatch(3C)、 glob(3C)、 regcomp(3C)、 setlocale(3C)、 environ(5)。

符合的标准

<regexp.h>：AES、SVID2、SVID3、XPG2、XPG3 和 XPG4

已过时

名称

region_hash_locks - 已过时的内核可调参数

说明

region_hash_locks 可调参数已过时并被删除。 HP-UX 将根据系统配置自动计算此值。

全局内核结构包含运行进程或内存使用的相关信息，经常被几个线程同时访问或修改。要防止发生争用，可使用 spinlock （用于进行同步的内核数据结构）对这些结构进行保护， spinlock 仅允许 spinlock“持有者”继续操作，而试图访问结构的所有其他用户都必须等待。

当需要保护此类数据结构的每个实例，而且有多个实例时，将使用散列的 spinlock。所有实例仅使用一个 spinlock 将导致过多的争用，但是每个结构使用一个 spinlock 将导致内存浪费，而且在任何给定时刻，大多数锁都没有被使用。

通过分配一个散列锁池，散列函数可为每组结构选择一个锁，从而减少了争用内存的情况，并节省了内存。系统计算的 **region_hash_locks** 值设定了这种区域数据结构 spinlock 池的大小。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

region_hash_locks 由 HP 开发。

名称

remote_nfs_swap - 启用跨 NFS 交换

值

缺省值

0 （关闭）

允许值

0 （关闭） 或 **1** （打开）

说明

此可调参数控制添加用作交换的 NFS 文件系统。如果 **remote_nfs_swap** 被设置为 **0** （关闭），那么只有本地文件系统和设备可用于交换。如果它被设置为 **1** （打开），那么本地和网络文件系统都可以用于交换。

以前，此可调参数用于现已不再支持的 NFS 群集中，但此功能仍然没有被删除。

应该由谁来更改此可调参数？

任何人。

对于更改的限制

对此可调参数的更改将在下次引导时生效。

何时应打开此可调参数？

几乎从不。如前所述，此可调参数设计用于那些今天不再使用的系统。仅拥有极其强大 NFS 功能的系统才应考虑使用 NFS 用于交换。

打开此可调参数的副作用是什么？

如果此值被设置为 **1** （打开）且添加了一个 NFS 分区作为交换，那么将对交换文件系统的 NFS 事务专门设置一些内核内存。然后内核会像使用本地文件系统一样使用 NFS 交换分区。如果系统的 NFS 功能不强大，则可能导致交换时间（交换入和交换出）极长，且可能导致内存的丢失因为导致交换需要的 NFS（仅当内存压力高时）可能使用所有保留及更多的内存。

何时应关闭此可调参数？

除非肯定 NFS 系统可以处理用作交换的负载，且没有本地文件系统或磁盘驱动器备用，否则此可调参数应 **always** 被设置为 **0** （关闭）。

关闭此可调参数的副作用是什么？

将不再允许任何先前定义的 NFS 交换文件系统。将不再为由于交换导致的 NFS 事务保留内核内存。

同时应更改哪些其他可调参数？

无。

警告

所有 HP-UX 内核可调参数都是特定于各个发行版的。未来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺

省值或建议的值。有关安装对可调参数值的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

remote_nfs_swap 由 HP 开发。

名称

rtsched_numpri - 支持 POSIX.1b 实时应用程序优先级值的数量

值

无故障

32

缺省值

32

允许值

允许在 **32** 到 **512** 范围内的任意值。 低于 **32** 的值被重置为 **32** ， 而高于 **512** 的值被重置为 **512** 。

建议值

32

说明

rtsched_numpri 可调参数定义了支持 POSIX 1.b 实时应用程序的优先级值的数量。在管理所有线程彼此之间以及所有线程与系统中运行的其他应用程序之间的相对优先级中，更大的值为应用程序提供了更多的灵活性。然而，由于需要管理可能会增加搜索时间的更大运行队列，更大的值增加了操作系统中处理的开销。

应该由谁来更改此可调参数?

任何人。

对于更改的限制

对此可调参数的更改将在下次重新引导时生效。

何时应增加此可调参数的值?

当系统主要运行 POSIX 实时应用程序，同时在管理应用程序相对优先级时需要更多的灵活性时。

增加此值的副作用是什么?

增加 **rtsched_numpri** 可调参数的值可能由于更大和可能更稀疏的运行队列导致一些操作系统开销。

何时应降低此可调参数的值?

rtsched_numpri 可调参数的缺省值已被设置为最小的可能值。

同时应更改哪些其他可调参数的值?

无。

警告

所有 HP-UX 内核可调参数都是特定于各个发行版的。未来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

rtsched_numpri 由 HP 开发。

名称

sched_thread_affinity - 调整调度程序线程关系

值

保证安全

6

缺省值

6

允许值

允许使用范围 **1** 到 **10** 的任意值。

值 **1** 表示较弱的线程关系，并且线程可在不同的处理器上运行。值 **10** 表示较强的线程关系，并且线程很可能保留在同一处理器上。

建议值

正常情况下选用缺省值。

在特殊情况下，如果必须使线程尽快运行，则可以使用接近 **1** 的值。但是，这可能会造成缓存数据丢失的增加。

客户在生产系统上更改此可调参数值之前，必须评估在其工作负荷环境中所造成的性能影响。

说明

sched_thread_affinity 可调参数定义线程可能在哪个处理器上运行。关系不紧密的线程在其生命周期内可以从一个处理器移至另一个处理器，而关系较紧密的线程在处理器之间的移动次数可能是最少的。

sched_thread_affinity 值发生的更改可能对系统吞吐量及响应时间产生直接影响。使用极小的值可能使线程的运行速度快很多，但也可能增加缓存数据丢失；如果线程已运行（或正在运行），则使用极大的值可能导致线程在加载的处理器上运行效率更低，但可能带来缓存关系提升的好处。

对此可调参数使用较小值，可能会使 I/O 密集型应用程序受益。

更改此可调参数的人员

任何用户。

更改限制

对此可调参数的更改立即生效。

应该何时增加此可调参数的值

只要增大此可调参数值，大多数线程在其生命周期内就会保留在其运行所在的服务器上。

增加此值的负面影响

增加 **sched_thread_affinity** 可调参数的值将导致某些线程等待运行，即使系统中存在较少的已加载处理器或空闲处理器，也是如此。

应该何时降低此可调参数的值

如果希望工作负荷尽快运行，而不管它是否在处理器之间移动，且可能会增加缓存数据丢失，则应降低 **sched_thread_affinity** 可调参数值。

降低此值的负面影响

降低 **sched_thread_affinity** 可能会导致缓存数据丢失增加，因为线程在其生命周期内将开始在不同的处理器上运行。

应该同时更改的其他可调参数值

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

sched_thread_affinity 由 HP 开发。

名称

scroll_lines - 由内置仿真终端使用的可滚动行数

值

无故障

100

缺省值

100

允许值

60 到 **999**

建议值

不大于足够用户使用的值。

说明

本可调参数指定了由 HP-UX 图形控制台内置仿真终端 (ITE) 使用的滚动缓冲行的总数。该值是屏幕上与屏幕外行数的总和。例如，如果 ITE 在屏幕上有 68 行，并且 scroll_line 的值为 128，则在屏幕外将有 60 行可以用于回滚查看。

在引导期间，系统可能会向上调整 **scroll_lines** 的值，这取决于所安装的图形硬件。如果图形卡可以在一种以上分辨率情况下运行，则 ITE 将首先确定在所有可能的分辨率情况下，屏幕上可显示的最大行数。然后，ITE 将确保 **scroll_lines** 至少等于屏幕上的行数。例如，假定图形卡支持两种可能的分辨率，在一种 ITE 分辨率的情况下为每行 160 个字符，共 62 行；在另一种 ITE 分辨率的情况下为每行 120 个字符，共 78 行。如果系统文件中已指定 **scroll_lines** 值为 62，则 ITE 会将其向上调整为 78。

应该由谁来更改此可调参数？

任何人。

对于更改的限制

对此可调参数的更改将在下次重新引导时生效。

何时应增加此可调参数的值？

如果用户希望在滚动缓冲区中保留其他的行，则可以增加 **scroll_line** 的值。

增加此可调参数值的副作用是什么？

增加 ITE 滚动缓冲区的大小会消耗内核内存，此内存空间将专用于 ITE，而不能由系统的其余部分用于其他目的。内存使用比率为每个屏幕外字符对应 2 个字节。

何时应降低此可调参数的值？

如果滚动缓冲区中不需要屏幕外的其他行，则可以减少 **scroll_lines** 的值，以释放少量内存。节省的内存量为每个屏幕外字符对应两个字节。例如，如果显示屏为 160 个字符宽，则 **scroll_lines** 减少 10，将释放 3200 个字节。

降低此可调参数值的副作用是什么？

如果 **scroll_lines** 值设置为小于或等于显示在 ITE 屏幕上的行数，则屏幕外文本将不可访问。

同时应更改哪些其他可调参数？

无。

警告

所有 HP-UX 内核可调参数都是特定于各个发行版的。未来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

scroll_lines 由 HP 开发。

名称

scsi_maxphys -所有 SCSI 设备上允许的最大 I/O 长度（已过时）

值

保证安全

1048576

缺省值

1024*1024

允许值

1048576

建议值

1048576

说明

注释：此可调参数已过时，并且已替换为 **escsi_maxphys** 属性，该属性可使用 **scsimgr** 命令来获取和设置（请参阅 *scsimgr*(1M)）。

此可调参数设置了 SCSI 子系统接受进行 I/O 的最大数据大小。依据设备特性及设备驱动程序配置，对于特定的 SCSI 设备，SCSI 子系统允许的最大大小可能小于或等于此可调参数的值。但决不会大于它的值。

更改此可调参数的人员

任何客户。

更改限制

对此可调参数的更改将在首次打开设备时生效。

应该何时增加此可调参数的值

绝不能增加此值。

应该何时降低此可调参数的值

绝不能降低此值。

更改此可调参数的同时应更改的其他可调参数

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。对于 HP-UX 11i v3 而言，此参数已过时。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

scsi_maxphys 由 HP 开发。

名称

scsi_max_qdepth - 目标排队等待执行的最大 I/O 数量（已过时）

值

保证安全

1

缺省值

8

允许值

1 - 255

建议值

1 - 255

大多数 SCSI-2 及更高级设备接受多个命令，并有足够内存支持由 HP 设置的缺省队列深度。用户可以更改缺省值以调整设备，来获得更高的处理能力或负载平衡。

说明

注释：此可调参数已过时并已替换为 **max_q_depth** 属性。该属性可通过 **scsimgr** 命令设置。请参阅 *scsimgr*(1M)。

某些 SCSI 设备支持标记队列，这意味着它们可以在任何时刻具有多个未完成的 SCSI 命令。未完成的命令数量随设备不同而变化，对于 HP-UX 是未知的。为避免溢出队列，当任何 SCSI 设备上的未完成命令超过一定数量时，HP-UX 将不再发送命令。此可调参数设置了缺省的限制值。对于特定设备，可以使用 **ioctl** 来覆盖此缺省值。

队列深度与标记队列意义相同。当目标支持它时，将允许目标接受多个等待执行的 SCSI 命令。某些目标可允许存储多达 256 个来自不同启动程序的命令。此机制有助于优化以获得更佳性能。一旦目标的命令队列排满，目标将终止任何其他 I/O，并将 **QUEUE FULL** 状态返回到启动程序。目标可能支持少于 256 个命令排队等待，因此出厂缺省值为 **8** 。

如果系统中存在支持小队列深度设备和支持较大队列深度设备的组合，则队列深度可设置为能用于大多数设备的值。对于特定设备，系统管理员可以使用 **SIO_SET_LUN_LIMIT ioctl()** 更改每个设备的队列深度。有关如何使用 **ioctl()** 的详细信息，请参阅 *scsictl*(1M) 。

对于 32 位和 64 位内核，取值是相同的。

更改此可调参数的人员

任何用户。

更改限制

对此可调参数的更改立即生效。

应该何时增加此可调参数的值

SCSI 设备具有足够内存，支持比 HP 设置的缺省值更高的队列深度时。如果队列深度设置为更高的值，则此类设备可以提供更高的性能。

增加此可调参数值的负面影响

队列深度适用于所有支持标记队列的 SCSI 设备。若队列深度设置值超出磁盘可处理的范围，则一旦磁盘上出现 **QUEUE FULL** 的情况，将导致 I/O 延缓。存在一种可在 **QUEUE FULL** 的情况下降低设备队列深度的机制，可以避免该设备上出现无限 **QUEUE FULL** 的情况。然而，此机制将定期尝试更高的队列深度，从而引发 **QUEUE FULL** 情况。

应该何时降低此可调参数的值

当所连接的 SCSI 设备支持更小的队列深度或者为了达到负载平衡的时候。

降低此可调参数值的负面影响

当队列深度设置为较低值时，支持更高队列深度的设备不可能提供最佳性能。

应该同时更改的其他可调参数

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。对于 HP-UX 11i v3 而言，此参数已过时。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

scsi_max_qdepth 由 HP 开发。

另请参阅

scsictl(1M)、 ioctl(2)、 scsi(7)。

名称

sd - Software Distributor，用于创建、分发、安装、监视和管理软件的命令

概要

sw<*command*> [*XToolkit Options*] [**-r**|**-d**] [**-i**] [**-l**] [**-p**] [**-R**] [**-u**] [**-v**] [**-V**]
[**-a** *attribute*] [**-c** *catalog*] [**-C** *session_file*] [**-D** *acl_entry*] [**-f** *software_file*] [**-F** *acl_file*]
[**-J** *jobid*] [**-l** *level*] [**-M** *acl_entry*] [**-Q** *date*] [**-s** *source*] [**-S** *session_file*]
[**-t** *target_file*] [**-x** *option=value*] [**-X** *option_file*] [*software_selections*] [@ *target_selections*]

备注

- 可以启用 Software Distributor (SD) 来管理远程系统上的软件。有关详细信息，请参阅下面的“远程操作”一节。
- 键入 **man 4 sd** 以查看 *sd*(4) 手册条目，了解所有 SD 对象、属性和数据格式的说明。
- 键入 **man 4 swpackage** 以查看 *swpackage*(4) 手册条目，了解用作 **swpackage** 命令输入的产品规范文件 (PSF) 的说明。

说明

有关 SD 的完整说明，请参阅《Software Distributor 管理指南》"（位于 **http://docs.hp.com/zh_cn/** ）。

SD 命令和相关程序：

- **sd** - 用于以交互方式创建、调度和监视软件作业和日志文件。还用于启动安装、复制和删除命令。
- **swacl** - 修改用于控制 SD 安全性的访问控制列表 (ACL)。
- **swagentd** - 用于为本地或远程 SD 软件管理任务提供服务并启动 SD 代理的守护程序。
- **swask** - 运行请求要在软件安装或配置中使用的用户响应的脚本。
- **swcluster** - 配置无盘客户端（仅限 HP-UX 10.X）。
- **swconfig** - 配置、取消配置或重新配置已安装软件。
- **swcopy** - 将软件产品复制到软件仓库，以便进行后续安装或分发。
- **swinstall** - 安装并配置软件产品。
- **swjob** - 创建并监视软件作业和日志文件。
- **swlist** - 显示有关软件产品的信息。
- **swmodify** - 在目标根目录或软件仓库中修改软件产品信息。
- **swpackage** - 将软件产品打包到分发目录或串行格式的软件仓库中。
- **swreg** - 注册或取消注册软件仓库或根目录。
- **swremove** - 删除并取消配置软件产品。

- **swverify** - 验证软件产品。
- **install-sd** - 从新介质中检索 SD 产品（以及所有相关的修补软件）并进行安装。

下列各节着重说明了这些命令支持的功能。

远程操作

可以启用 Software Distributor (SD) 来管理远程系统上的软件。要允许中央 SD 控制器（也称为 中央管理服务器或管理器节点）中的根用户对远程 目标（也称为 主机或代理）执行操作，请执行以下操作：

1) 在远程计算机上设置超级用户、主机和模板访问控制列表 (ACL)，以允许从控制器系统访问根目录。要执行此操作，请在每个远程系统上运行以下命令：

/usr/lib/sw/mx/setaccess *controller*

注释：

- *controller* 是中央管理服务器的名称。
- 如果远程系统是 11.00，请确保在运行 **setaccess** 之前，已在该远程系统上安装了 SD 修补软件 PHCO_22526 或者替代修补软件。
- 如果远程系统的版本比 11.00 旧，或者因为其他原因使得 **setaccess** 不正确，则从 11.11 或更高版本的系统将 **setaccess** 脚本复制到此远程系统。

2) **swinstall**、 **swcopy** 和 **swremove** 具有用于远程操作的增强的 GUI 界面。通过在控制器上创建 **.sdkey** 文件，可以启用这些增强的 GUI 界面。请使用以下命令：

touch /var/adm/sw/.sdkey

注释：还可以通过在远程计算机上直接使用 **swacl** 命令，来向控制器系统中的用户授予超级用户访问权限或非超级用户访问权限。

交互式操作

缺省情况下，除 **sd** 和 **swask** 以外的所有 SD 命令都在非交互模式下操作。 **swcopy** 、 **swinstall** 、 **swlist** 和 **swremove** 命令也支持图形用户界面 (GUI)（如果终端或显示设备不支持 GUI，这些命令还提供了终端用户界面，可以使用键盘在其中执行屏幕导航操作，而无需使用鼠标）。

要调用该 GUI，请输入不包含任何命令行选项的命令，或在调用该命令时添加 **-i** 选项以及其他命令行选项。 必须指定 **-i** 选项才能调用 **swlist** GUI。

如果将 **ask** 选项设置为 **true** ， **swconfig** 和 **swinstall** 的命令行版本将以交互方式运行。该选项执行交互式 请求脚本。

sd 命令是一个用于监视和调度软件作业的交互式界面。该命令与 **swjob** 命令提供相同的功能。还可以使用 **sd** 来调用 **swinstall** 、 **copy** 和 **swremove** GUI。

如果启用了 SD 的远程操作功能， **swinstall** 、 **swcopy** 和 **swremove** 将提供增强型 GUI，以支持对远程目标执行操作。有关启用远程操作和增强型 GUI 的详细信息，请参阅上面的 远程操作。

分布式操作

除 **swask** 、 **swpackage** 和 **swmodify** 以外的所有 SD 命令都使用分布式操作模型。这些命令用作分布式操作的控制器，管理特定的软件管理任务。对于每个 *target_selection* ， SD 代理进程将执行以下任务：

- **swagent** - 作为 SD 命令的代理执行软件管理任务。

命令与每个代理之间的通信，以及其他目标主机活动均由 SD 守护程序进程实现：

- **swagentd** - 为本地或远程软件管理任务提供服务。

软件作业管理

swinstall、**copy** 和 **remove** 命令用于创建作业信息，以记录作业定义（在会话文件中）、状态以及作业的日志信息。用户可以立即执行作业，也可以调度作业以便以后执行。可以使用 **swjob** 命令或 **sd** 交互式界面浏览调度作业、活动作业以及完成的作业。

安全操作

SD 使用访问控制列表 (ACL) 授权用户尝试创建、修改或读取软件仓库中的软件产品，或安装到根文件系统的软件产品。超级用户可以向特定的本地和远程用户授予对目标主机、目标软件仓库和（或）目标根文件系统的特定访问权限（请注意，SD 不使用 ACL 来执行本地超级用户调用的任务）。

由于要以超级用户身份加载文件并运行脚本，因此通过授予对根文件系统的写入权限（以安装软件）或对主机的插入权限（以创建新的根目录），可以向用户有效授予超级用户权限。

SD 使用基于凭据和口令的方法来验证用户以及执行给定操作的 SD 命令。

SD 还具有一种非特权模式，该模式将 ACL 授权替换为用户文件权限。有关详细信息，请参阅 **run_as_superuser** 缺省选项和《Software Distributor 管理指南》。

灵活的策略控制

可以使用命令 缺省选项来控制 SD 命令的多个策略和行为。可以在系统级或用户特定的 SD 缺省值文件中定义这些选项，在调用命令时在命令行指定这些选项，或在 UI 中指定选定的选项。有关详细信息，请参阅下面的“缺省选项”标题。

预览、诊断和日志记录

除 **swlist** 和 **swjob** 以外的所有命令都会记录控制器主机上的主要事件以及目标主机上的详细事件。

如果源计算机和目标计算机均正在运行 HP-UX 11.00 或更高版本，则源软件仓库计算机上的系统管理员可以跟踪哪个用户从源计算机上的软件仓库中取出了 哪个软件以及 何时取出软件。有关详细信息，请参考 *swagent*(1M) **source_depot_audit** 选项。

可以使用 SD 交互式界面（使用 **sd** 命令进行调用）和 **swjob** 命令行界面来监视作业进度并查看控制器和目标日志文件。

swconfig 、 **swcopy** 、 **swinstall** 、 **swmodify** 、 **swpackage** 和 **swremove** 命令支持预览模式，在该模式下，这些命令将经过分析阶段，然后退出。

软件产品

软件产品采用多级的层次结构进行组织： 软件包、 产品、 子产品和 文件集。组成产品的实际文件将打包到文件集中。 SD 命令的 *software_selections* 可以指定软件包、产品、单个子产品和（或）单个文件集。

兼容的软件

软件产品指定它们支持（即兼容）的计算机类型和操作系统。 **swconfig** 、 **swinstall** 和 **swverify** 命令可以检测和（或）强制使用兼容的软件。

供应商定义的属性

可以在打包软件时创建用户自己的软件属性。对于产品规范文件中无法由 SD 识别的关键字及其关联值，可以将其传输到生成的 INDEX 或 INFO 文件（由 **swpackage**或 **swcopy** 创建）进行保留（有关 INDEX 和 INFO 文件的详细信息，请参考 *swpackage*(4) ）。

打包期间或使用 **swmodify** 进行修改时，系统不会记录供应商定义的属性。可以使用 **swlist** 列出这些属性。

软件之间的相关性

swask 、 **swconfig** 、 **swcopy** 、 **swinstall** 、 **swremove** 和 **swverify** 命令支持 相关性，即在安装其他软件之前或安装期间必须存在或缺少的软件。相关性可适用于文件集之间以及文件集与产品之间。SD 支持三种类型的相关性： 必备条件，必须在（分别）安装和配置相关文件集之前先进行安装和配置； 共存条件，必须在使用相关性之前进行安装和配置。 互斥条件，防止安装或配置相关文件集（如果存在）。

如果 *software_selection* 针对其他文件集和（或）产品指定了相关性，则命令将自动选择该软件。但 **swremove** 除外，该命令可以自动选择相关软件（依赖于 *software_selections* 的文件集和（或）产品）。

缺省情况下，必须先解析所有相关性，然后命令才能继续操作。

请注意，如果为文件集指定了相关性，且该文件集被作为修补软件一部分的其他文件集取代，则 SD 仍可以识别该相关性。

产品位置和多个版本

swinstall 命令可以将软件产品安装到备用产品位置，而不是安装到供应商指定的缺省产品目录。（此目录位置是所有产品文件的根目录）。

swinstall 命令还可以将多个版本的软件产品安装到单个目标系统，每个版本位于唯一的产品位置中。

软件管理命令 **swconfig** 、 **swlist** 、 **swremove** 和 **swverify** 允许用户通过指定作为 *software_selection* 一部分的产品位置，从多个已安装的版本中选择特定产品。

备用根目录和软件仓库目录

缺省情况下， **swinstall** 、 **swlist** 、 **swmodify** 、 **swremove** 和 **swverify** 命令对目标主机的主根文件系统 **(/")** 执行操作。这些命令允许用户使用 **@** *target_selection* 语法和 **-r** 命令行选项指定备用根目录（此选项不是必需的，保留它的主要目的是实现向后兼容）。

注释：

- 备用根目录是缺省主根目录 (/) 以外的根文件系统（备用根目录最终将成为目标主机的根目录）。

- 对备用根目录执行的操作不包括兼容性过滤。
- 对于针对备用根目录执行的操作，将不运行配置、取消配置和验证脚本。
- 安装期间不能使用此选项重定位软件。必须在软件选择组件中使用 **l=location** 语法。
- 备用根目录操作并不等效于 **chroot** 命令。

备用根目录为某些测试环境提供了好处（如通过挂接其根文件系统来构建测试系统）。还可以使用它们将软件仓库中的文件快速置于系统中，以便进行查看或用于其他目的。

对软件仓库执行操作时，缺省情况下， **swcopy** 、 **swpackage** 、 **swlist** 、 **swverify** 、 **swremove** 和 **swverify** 命令将使用位于 **/var/spool/sw** 中的软件仓库。用户还可以指定这些命令的备用软件仓库目录。

磁盘空间分析

swcopy 、 **swinstall** 和 **swpackage** 命令对 *target_selections* 执行磁盘空间分析，确保有足够的可用磁盘空间用来执行任务。

对软件进行打包时，可以为文件集定义 空间文件，从而定义所需的额外空间（执行磁盘空间分析时，要考虑到空间文件）。

执行任何磁盘空间分析之前， **swcopy** 、 **swinstall** 和 **swpackage** （还有 **swverify** 和 **swremove** ）将执行 *mount*(1M) 命令，以挂接每个目标文件系统表（ /etc/fstab 或等效目录）中列出的所有文件系统。这样可确保不会将文件加载到未来挂接点之下的目录中。可以使用 **mount_all_filesystems** 选项覆盖此挂接策略。

控制脚本

swask 、 **swconfig** 、 **swinstall** 、 **swremove** 和 **swverify** 命令可以执行供应商定义的控制脚本，以便执行检查或执行通常使用这些命令所执行任务以外的其他任务。

一般情况下，SD 使用带有产品或文件集对象的脚本。脚本通常不会将 HP 制造的软件在出厂时就附带于新的系统中。

SD 支持以下类型的脚本：

- **Checkinstall** -（适用于 **swinstall** ）。检查脚本，用于分析要安装的每个 *target_selection* （目标主机），以确定是否可执行安装和配置。
- **Preinstall** -（适用于 **swinstall** ）。在安装软件文件以执行其他文件安装操作（如删除已过时的文件）之前立即执行的脚本。
- **Unpreinstall** -（适用于 **swinstall** ）。当 SD 在安装过程中必须启动恢复时执行的“撤销”安装前脚本。
- **Postinstall** -（适用于 **swremove** ）。在安装文件集或产品以执行其他删除操作（如重置缺省文件）后立即执行的脚本。
- **Unpostinstall** -（适用于 **swremove** ）。当 SD 必须在安装过程中启动恢复时执行的“撤销”安装后脚本。

- **Configure** -（适用于 **swconfig** 、 **swinstall** 和 **swremove** ）。用于配置已安装文件集或产品的脚本。
- **Unconfigure** -（适用于 **swconfig** 和 **swremove** ）。配置脚本为“取消”配置而执行的脚本。
- **Verify** -（适用于 **swverify** ）。用于验证文件集或产品配置的脚本。（该脚本除了执行标准的 **swverify** 检查以确认文件是否与 SD 数据库条目一致以外，还将执行上述配置检查）。
- **Checkremove** -（适用于 **swinstall** ）。检查脚本，用于在删除之前先分析每个 *target_selection* （目标主机），以确定是否可以执行删除和取消配置。
- **Preremove** -（适用于 **swremove** ）。在删除软件文件以执行其他文件操作（如删除安装前脚本创建的文件）之前立即执行的脚本。
- **Postremove** -（适用于 **swremove** ）。删除文件集或产品以执行其他删除操作（如恢复“回滚”文件）后立即执行的脚本。
- **Request** -（适用于 **swask** 、 **swconfig** 和 **swinstall** ）。在安装或配置过程中从用户那里请求响应的交互式脚本。
- **Other scripts** -可以将其他专用的脚本作为子脚本包含在标准的 SD 控制脚本中。

有关使用控制脚本的详细信息，请参阅《Software Distributor 管理指南》。

软件状态

SD 命令可在各种 状态之间来回转换产品和文件集。

安装期间，软件将在以下状态之间来回转换：

- 不存在
- 瞬态
- 已安装
- 已配置

删除期间，软件将在以下状态之间来回转换：

- 已配置
- 已安装
- 瞬态
- 不存在

对软件进行打包或将软件复制到软件仓库中时，软件将在以下状态之间来回转换：

- 不存在
- 瞬态
- 可用

从软件仓库中删除软件时，软件将在以下状态之间来回转换：

- 可用
- 瞬态

- 不存在

如果任务在处于瞬态状态期间失败，则状态将被设置为损坏。

选项

下列选项受一个或多个 SD 命令支持。有关特定于该命令的选项，请参考与每个命令对应的手册页。

XToolKit 选项
: 交互式命令支持用于控制 GUI 外观的标准 X Toolkit 选项的子集。支持的选项如下：**-bg** 、 **-background** 、 **-fg** 、 **-foreground** 、 **-display** 、 **-name** 、 **-xrm** 以及 **-synchronous** 。有关这些选项的定义，请参阅 *X*(1) 联机帮助页。

-d
: 可使命令对 *target_selections* （软件仓库，而不是根目录）执行操作。

-r
: 可使 SD 命令对备用根目录（必须在 @ *target_selections* 选项中指定）执行操作（该选项并不是备用根目录操作的必需选项，而是为了实现向后兼容性而保留的。

  有关详细信息，请参阅上面的“备用根目录和软件仓库目录”标题）。

-i
: 在交互模式（图形用户界面）下运行命令。有关其他详细信息，请参阅上面的 交互式操作和 远程操作标题。

-l
: （仅限 **HP-UX 10.X**）在 **linkinstall** 模式下运行命令，使安装在服务器 共享根目录下的软件可用于无盘客户端的 专用根目录。

  在 *linkinstall* 模式下运行时， **swinstall** 可以：

  - 创建软件的 NFS 挂接，使其可从目标中得到访问。这可能需要延迟备用根目录的挂接。
  - 修改目标的 *fstab* 文件。
  - 修改源的 **exports** 文件以添加对目标的挂接权限。

  可以通过检查 *share_link* 产品属性来创建挂接。并非所有产品都支持 **linkinstall** 。如果某些产品位于现有挂接下，则无需创建新的挂接即可看到它们。

-p
: 通过在分析阶段执行会话并在该命令开始执行实际任务之前退出的方式预览任务。

-R
: 对于 **swlist** ，该命令可采用递归方式将所有对象纳入文件集级别。
  对于 **swjob** ： 该命令可采用递归方式将所有对象纳入 *end_target* 级别。

-u
: 撤消操作差异，使用 **swconfig** 取消配置软件、使用 **swreg** 取消注册指定的对象或使用 **swjob** 命令删除指定的作业。

-v
: 打开详细输出，以输出到 stdout（命令日志文件不受此选项的影响）。缺省情况下，将对所有 SD 命令启用详细输出。

-V
: 列出支持的数据模型修订版。

-a *attribute* 指定要使用 **swlist** 、 **swmodify** 或 **swjob** 命令显示或修改的特定属性。

-c *catalog* 指定包含导出清单的目录的路径名。对于 **swask** ，此清单存储请求脚本创建的响应文件的副本。对于 **swlist** 和 **swmodify** ，此清单存储这些命令的输出或输入。

-C *session_file* 将当前选项和操作数保存至 *session_file* （可以使用 **-S** *session_file* 选项重新调用会话文件）。有关详细信息，请参阅本联机帮助页中的 会话文件标题。

-D *acl_entry* 使用 **swacl** 删除与指定对象关联的 ACL 中的现有条目。

-f *software_file* 从 *software_file* 而不是命令行（或不仅是命令行）读取 *selections* 的列表。

-F *acl_file* 使用 **swacl** 将 *acl_file* 中包含的 ACL 分配给指定对象。

-J *job_id* 执行以前调度的作业。 **swagentd** 使用此选项来启动调度的作业。

-l *level* 在使用 **swlist** 时列出指定 *level* 的所有对象，或者在使用 **swacl** 或 **swreg** 时定义对象的级别。

-M *acl_entry* 使用 **swacl** 添加新的 ACL 条目或更改现有条目的权限。

-Q *date* 针对指定的日期和时间调度命令。

-s *source* 指定从中安装、复制、列出或打包软件的源软件仓库、PSF 文件或磁带。（SD 可以读取 **tar** 和 **cpio** 磁带软件仓库）。

-S *session_file* 基于 *session_file* 中的前一个会话保存的选项和操作数执行命令（可以使用 **-C** *session_file* 选项将会话信息保存到文件）。有关详细信息，请参阅本联机帮助页中的会话文件标题。

-t *target_file* 从 *target_file* 而不是命令行（或不仅是命令行）读取 *target_selections* 的列表。

-x *option=value* 将会话 *option* 设置为 *value* 并覆盖缺省值（或使用 **-X** 选项指定的备用 *option_file* 中的值）。可以指定多个 **-x** 选项。

-X *option_file* 从 *option_file* 中读取会话选项和行为。在此文件中定义的这些值将覆盖缺省值。

操作数

大多数 SD 命令支持两种类型的操作数： “软件选择”后跟 “目标选择”。这些操作数之间用“at” (@) 字符分隔。此语法表示对“在目标位置的选择”执行命令。

软件选择

selections 操作数由大多数 SD 命令的 *software_selections* 组成。对于 **swjob** 和 **swreg** 命令，选择可分别为 *job_ids* 和 *roots_or_depots* 。

对于每个 *software_selection* ， SD 命令支持以下语法：

bundle[**.***product*[**.***subproduct*][**.***fileset*]][**,***version*]

product[**.***subproduct*][**.***fileset*][**,***version*]

- **=** （等号）关系运算符允许使用以下 Shell 通配符和模式匹配表示法指定选择：

  [ ], *, ?

  例如，以下表达式可安装带有以“man”结尾的标记的所有软件包和产品：

  swinstall -s sw_server *man

- 软件包和 子产品是递归的。 软件包可以包含其他 软件包，而 子产品可以包含其他 子产品。例如：

  swinstall bun1.bun2.prod.sub1.sub2.fset,r=1.0

  或（使用表达式）：

  swinstall bun[12].bun?.prod.sub*,a=HP-UX

- ***** 软件规范可以选择所有产品。使用此规范时请务必小心。

version 组件格式如下：

[**,r** *<op> revision*][**,a** *<op> arch*][**,v** *<op> vendor*]
[**,c** *<op> category*][**,q=***qualifier*][**,l=***location*]
[**,fr** *<op> revision*][**,fa** *<op> arch*]

- *location* 仅适用于已安装软件，并引用安装到非缺省产品目录位置的软件。

- **fr** 和 **fa** 仅适用于文件集。

- **r** 、 **a** 、 **v** 、 **c** 和 **l** 仅适用于软件包和产品。它们适用于软件规范中最左侧的软件包或产品。

- *<op>* （关系运算符）组件的格式如下：

  = 、 **==** 、 **>=** 、 **<=** 、 **<** 、 **>** 或 **!=**

  对用点分隔的字段执行单独的比较。

  例如， **r>=B.11.00** 选择高于或等于 **B.11.00** 的所有修订版。系统将比较每个用点分隔的字段以查找匹配。不允许将 Shell 模式用于这些运算符。

- **=** （等号）关系运算符允许使用以下 Shell 通配符和模式匹配表示法指定选择：

  [ ] 、 ***** 、 **?** 、 **!**

  例如，表达式 **r=1[01].*** 返回第 10 版或第 11 版中的任何修订版。

- 所有版本部分在单个规范中是可重复的（例如 **r>=A.12** 、 **r<A.20** ）。如果使用了多个部分，则选择必须匹配所有部分。

- 完全限定的软件规范包括 **r=** 、 **a=** 和 **v=** 版本部分，即使它们包含空字符串。对于已安装软件，还包括 **l=** 。

- 软件选择中不允许使用空格或制表符。

- 软件 **instance_id** 可代替版本部分。其格式如下:

  [*instance_id*]

  在已导出清单环境中，其中 **instance_id** 是一个整数，该整数可区分具有相同标记的产品和软件包的版本。

目标选择

对于每个 *target_selection* ，SD 命令支持以下语法。

[*host*][**:**][**/***directory*]

如果既指定了主机又指定了目录，则需要使用冒号 (**:**) 。

外部输入和语言环境影响

缺省选项

除了标准选项之外，可以通过编辑以下位置中的缺省值来更改多个 SD 行为和策略选项:

/var/adm/sw/defaults	系统级缺省值。
$HOME/.swdefaults	用户特定的缺省值。

必须在缺省文件中使用以下语法指定值:

[*command_name***.**]*option***=***value*

可选的 **command_name** 前缀表示 SD 命令之一。如果使用此前缀，则对缺省值的更改将仅应用于该命令。如果不使用此前缀，则更改将应用于所有命令。

还可以在命令行中使用 **-x** 或 **-X** 选项覆盖缺省值:

command **-x** *option***=***value*

command **-X** *option_file*

下面一节列出 SD 命令支持的所有关键字。每个命令的联机帮助页还列出了该命令支持的关键字。如果存在缺省值，则列在 **=** 之后。此外，还指定了此选项适用的命令。

admin_directory=/var/adm/sw（用于普通模式）
admin_directory=/var/home/LOGNAME/sw（用于非特权模式）

SD 日志文件的位置和已安装软件清单的缺省父目录。对于普通 SD 操作，缺省值是 **/var/adm/sw** 。当 SD 以非特权模式操作（也就是说，当将 **run_as_superuser** 缺省选项设置为 **true** ）时:

- 缺省值会强制设置为 **/var/home/LOGNAME/sw** 。
- 路径元素 **LOGNAME** 将被替换为调用用户的名称，这是 SD 从系统口令文件中读取的。
- 如果将此选项的值设置为 **HOME/***path* ，则 SD 将用调用用户的主目录（来自系统口令文件）替换 **HOME** 并相对于该目录解析 路径。例如， **HOME/my_admin** 将解析为主目录

中的 **my_admin** 目录。

- 如果将 **installed_software_catalog** 缺省选项的值设置为相对路径，则将相对于此选项的值解析该路径。

SD 的非特权模式仅用于管理经过特殊设计和打包的应用程序。此模式不能用于管理 HP-UX 操作系统或其修补软件。有关非特权 SD 的完整说明，请参阅《Software Distributor 管理指南》（位于 **http://docs.hp.com/zh_cn/** 网站）。

另请参阅 **installed_software_catalog** 和 **run_as_superuser** 选项。

适用于除 **swagent** 、 **swagentd** 和 **install-sd** 以外的所有 SD 命令。

agent=/usr/lbin/swagent

守护程序调用的代理程序所在的位置。

适用于 **swagentd** 。

agent_auto_exit=true

可使目标代理在执行阶段后或分析阶段失败后自动退出。当控制器使用交互式 UI 时或当使用 **-p** （预览）时，此选项将强制为 **false** 。这可以增强网络可靠性和性能。缺省值 **true** 表示目标代理将在适当的时候自动退出。如果设置为 **false** ，则目标代理在控制器结束会话时才会退出。

适用于 **swconfig** 、 **swcopy** 、 **swinstall** 、 **swremove** 、 **swverify** 。

agent_timeout_minutes=10000

如果目标代理在指定时间内一直处于不活动状态，将使其退出。该选项可用于使目标代理更快地检测到中断的网络连接，这是因为 RPC 检测中断连接的时间长达 130 分钟。建议的值是环境中预期为最长的不活动的时间段。对于命令行调用， 10 分钟到 60 分钟之间的值比较合适。如果使用 GUI，则推荐值为 60 分钟或更长时间。缺省值 10000 略小于 7 天。

适用于 **swcopy** 、 **swinstall** 、 **swjob** 、 **swlist** 、 **swremove** 、 **swverify** 。

allow_downdate=false

防止安装目标中已存在的文件集的早期修订版（许多软件产品不支持“downdate”）。如果设置为 **true** ，则可以安装早期修订版。

适用于 **swinstall** 。

allow_incompatible=false

要求正在安装的软件产品与目标选择“兼容”（所有目标选择必须与为每个选定产品定义的支持系统的列表相匹配）。如果设置为 **true** ，则不会强制执行目标兼容性。

适用于 **swconfig** 、 **swinstall** 和 **swverify** 。

allow_multiple_versions=false

防止当在目标中已安装或已配置某一版本的情况下再安装或配置另一个独立的产品版本。

如果设置为 **true** ，则可以将现有产品的另一个版本安装到一个新的位置，或者可以在其新位置进行配置。只有在可以定位产品的情况下，才能安装多个版本。只有在产品支持多个已配置版本的情况下，这些版本才能正常工作。

适用于 **swconfig** 、 **swinstall** 和 **swverify** 。

allow_split_patches=false

允许使用单个修补软件文件集，而无需“同级”文件集。在缺省状态 **false** 下，当安装、复制或删除多文件集修补软件中的单个文件集时，将根据目标文件集的祖先文件集自动包括属于该修补软件的任何其他文件集（此行为适用于用户直接选择的文件集，以及 SD 解析软件相关性时自动选择的文件集）。

如果设置为 **true** ， SD 将允许安装、复制或删除单个修补软件文件集，而不包括同级文件集。这允许目标包含已“拆分”成其组件文件集的修补软件。警告：拆分修补软件可能会导致以下情况：同级组中的一个文件集将通过修补软件更新或删除，而其他文件集仍为早期版本或无法删除。

适用于 **swinstall**、**swcopy** 和 **swremove** 。

alternate_source=

定义将 **use_alternate_source** 选项设置为 **true** 时，代理将使用的替代源。替代源是使用以下语法指定的：

[*host*][**:**][*path*]

如果未指定 host 部分，将使用本地主机。如果未指定 path 部分，则将使用该命令发送的路径。代理尝试与替代源软件仓库联系时，将使用选项 **swagent.rpc_binding_info** 指定的协议序列和端点。

适用于 **swagent** 。

ask=true （仅限 **swask**）

ask=false （**swconfig** 和 **swinstall**）

执行要求用户响应的 请求脚本。如果 **ask=as_needed** ，则仅当控制目录中不存在响应文件时， **swinstall** 才会执行请求脚本。有关请求脚本的详细信息，请参阅 *swask*(1M) 。

适用于 **swask** 、 **swconfig** 和 **swinstall** 。

auto_kernel_build=true

通常设置为 true。指定删除内核文件集后是否应该重建内核。如果内核重建成功，系统会自动重新引导。如果设置为 false，系统将继续运行当前内核。

如果将 **auto_kernel_build** 选项设置为 **true** ，则必须将 **autoreboot** 选项也设置为 **true** 。如果将 **auto_kernel_build** 选项设置为 **false** ，则 **autoreboot** 选项的值将不起作用。

仅适用于 **swremove** 。

autoreboot=false

防止安装或删除需要从非交互式界面重新引导的软件。如果设置为 **true** ，则可以安装或删除软件，之后目标系统会自动重新引导。

交互式会话将始终要求用户在安装或删除需要重新引导的软件之前先进行确认。

如果将 **auto_kernel_build** 选项设置为 **true** ，则必须将 **autoreboot** 选项也设置为 **true** 。如果将 **auto_kernel_build** 选项设置为 **false** ，则 **autoreboot** 选项的值将不起作用。

适用于 **swinstall** 和 **swremove** 。

autorecover=false

此选项允许在出现安装错误时自动恢复原始文件集。但会导致磁盘空间暂时增加以及性能降低。缺省值 **false** 可使 **swinstall** 在更新文件集时删除原始文件。如果在安装期间出现错误（例如 网络故障），则原始文件将会丢失，用户必须重新安装文件集。

如果设置为 **true** ，则所有文件都会被另存为备份副本，直到当前文件集完成加载为止。如果在安装期间出现错误，则文件集的原始文件将被替换，而 **swinstall** 将继续处理产品中的下一个文件集或产品 **postinstall** 脚本。

如果设置为 **true** ，则此选项还将影响脚本。例如，如果 *preinstall* 脚本失败，则此选项使相应的 *unpreinstall* 脚本执行。有关完整的信息，请参阅《Software Distributor 管理指南》。

仅适用于 **swinstall** 。

autorecover_product=false

此选项允许在出现安装错误时自动恢复原始产品文件。但会导致磁盘空间暂时增加以及性能降低。缺省值 **false** 可使 **swinstall** 在更新产品集时删除所有现有的产品文件。如果在安装期间出现错误（例如网络故障），则原始文件将会丢失，用户必须重新安装产品。

如果设置为 **true** ，则所有产品文件都会被另存为备份副本，直到整个产品完成加载为止。然后，这些文件将被删除。如果安装期间出现错误，则原始产品文件将被替换，且 **swinstall** 将退出。

如果设置为 **true** ，则此选项还将影响脚本。例如，如果 *preinstall* 脚本失败，则此选项使相应的 *unpreinstall* 脚本执行。有关完整的信息，请参阅《Software Distributor 管理指南》。

仅适用于 **swinstall** 。

autoremove_job=false

控制已完成作业的自动作业删除。如果自动删除了作业，则不能使用 **swjob** 来查询作业信息（作业状态或目标日志文件）。

autoselect_dependencies=true

控制 SD 对必备软件、共存软件和互斥软件的自动选择。如果设置为 **true** ，则将自动选择必备软件进行配置。如果设置为 **false** ，则不会自动选择未明确选择的必备软件进行配置。如果设置为 **as_needed** ，则仅当目标不满足相关性时，才执行自动选择相关性操作。

适用于 **swconfig** 、 **swcopy** 、 **swinstall** 和 **swverify** 。

autoselect_dependents=false

控制 SD 是否自动选择相关软件。相关文件集已针对选定文件集建立了必备条件、共存条件或互斥条件。指定 **true** 将使得 SD 自动选择相关软件。缺省值 **false** 可阻止 SD 自动选择相关软件。

适用于 **swconfig** 和 **swremove** 。

autoselect_patches=true

为用户针对 **swinstall** 或 **swcopy** 操作选择的软件对象自动选择最新的修补软件（基于替换属性和祖先属性）。如果设置为 **false** ，则不会自动选择与选定对象对应的修补软件。

patch_filter 选项可与 **autoselect_patches** 结合使用。

适用于 **swask** 、 **swinstall** 和 **swcopy** 。

autoselect_reference_bundles=true

如果为 **true** ，则将自动安装或复制 粘着软件包，以及构成此软件包的软件。如果为 **false** ，则可以安装或复制软件，而不会自动包括包含软件的 粘着软件包。

对于 **swremove** ，如果设置为 **true** ，则在删除 is_sticky 属性设置为 true 的所有软件包的最近内容时，将自动删除这些软件包。如果设置为 **false** ，则不会自动删除粘着软件包。

适用于 **swcopy** 、 **swinstall** 和 **swremove** 。

check_contents=true

可使 **swverify** 验证文件的时间戳、大小和校验和属性。如果设置为 **false** ，则不会验证这些属性。

适用于 **swverify** 。

check_contents_uncompressed=false

（如果将 **check_contents** 设置为 **false** ，则将忽略此选项）。控制 **swverify** 是否验证压缩文件的大小和校验和。在缺省状态 **false** 下，**swverify** 只检查压缩文件的修改时间、大小和校验和属性。如果设置为 **true**，**swverify** 将在内存中解压缩文件，并验证解压缩内容的大小和校验和属性。

执行 **swverify** 操作期间，只能解压缩使用 SD 的内部压缩程序压缩的文件。有关详细信息，请参阅 **swpackage(1M)** 命令的 **compress_files** 选项。

适用于 **swverify** 。

check_contents_use_cksum=true

（如果将 **check_contents** 设置为 **false** ，则将忽略此选项）。控制 **swverify** 是否针对文件内容计算校验和。在缺省状态 **true** 下， **swverify** 检查包括校验和在内的所有文件属性。如果设置为 **false** ，则 **swverify** 仅检查文件时间戳和大小。

适用于 **swverify** 。

check_permissions=true

可使 **swverify** 验证已安装文件的模式、所有者、UID、组和 GID 属性。如果设置为 **false** ，则不会验证这些属性。

适用于 **swverify** 。

check_requisites=true

可使 **swverify** 验证是否满足了软件选择的必备条件、共存条件和互斥条件相关性。如果设置为 **false** ，则不会执行这些检查。

适用于 **swverify** 。

check_scripts=true

可使 **swverify** 对已安装软件运行文件集/产品验证脚本。如果设置为 **false** ，则不会执行这些脚本。

适用于 **swverify** 。

check_volatile=false

可使 **swverify** 不验证标记为易失性（即可更改）的那些文件。如果设置为 **true** ，则还会检查易失性文件（对于已安装软件）。

适用于 **swverify** 。

codeword=

提供对受保护 HP CD-ROM 软件进行解锁所需的“代码字”。

CD-ROM 所附带的部分 HP 软件产品为“受保护”产品。也就是说，如果不提供“代码字”和“客户 ID”，则无法安装或复制这些软件产品。代码字可在从 HP 收到的 CD-ROM 证书上找到。此选项用于存储代码字以备将来参考；代码字只需输入一次。

compress_cmd=/usr/contrib/bin/gzip

定义在安装、复制或打包之前压缩文件而调用的命令。如果将 **compression_type** 选项设置为 **gzip** 或 **compress** 以外的其他值，则必须更改此路径。

适用于 **swpackage** 和 **swagent** 。

compress_files=false

如果设置为 **true** ，则从源传输解压缩文件之前将对其进行压缩。这将针对 **swcopy** 和 **swinstall** 增强速度较慢的网络的性能，且会针对 **swcopy** 和 **swpackage** 生成较小的软件仓库（除非同时将 **uncompress_files** 选项设置为 **true** ）。

适用于 **swcopy** 、 **swinstall** 和 **swpackage** 。

compress_index=false

确定在写入目标软件仓库或根目录时， SD 命令是否会创建压缩的 INDEX 和 INFO 清单文件。如果设置为缺省值 **false** ，则不会创建压缩文件。如果设置为 **true** ，则 SD 将创建压缩和解压缩的 INDEX 和 INFO 文件。压缩文件名为 **INDEX.gz** 和 **INFO.gz** ，且与解压缩文件位于相同的目录中。

压缩文件可以提高速度较慢的网络的性能，尽管它们可能会由于较大的安装产品数据库和软件仓库清单而增加磁盘空间的使用。HP-UX 11.01 和更高版本的 SD 控制器和目标代理将在下列情况下从源代理自动加载压缩的 INDEX 和 INFO 文件：

- 源代理支持此功能。
- **INDEX.gz** 或 **INFO.gz** 存在于源软件仓库中。
- **INDEX.gz** 或 **INFO.gz** 不比相应的解压缩 INDEX 或 INFO 文件旧。

如果在访问、传输或解压缩 **INDEX.gz** 或 **INFO.gz** 文件时出现问题，则解压缩的 INDEX 或 INFO 文件将由源代理进行访问。

适用于 **swinstall**、**swcopy**、**swpackage** 、 **swmodify**、**swconfig** 和 **swremove** 。

compression_type=gzip

定义代理在传输文件期间或传输之后压缩这些文件时所使用的缺省压缩类型。如果将 **uncompress_files** 设置为 false，则将记录每个压缩文件的 **compression_type** ，以便稍后在执行 **swinstall** 或 **swcopy** （将 **uncompress_files** 设置为 true）期间应用正确的解压缩设置。指定的 **compress_cmd** 必须生成指定了 **compression_type** 的文件。 **uncompress_cmd** 必须能够处理指定 **compression_type** 的文件，除非格式为 **gzip** ，该格式由内部解压缩程序 (**funzip**) 进行解压缩。

适用于 **swagent** 。

config_cleanup_cmd=/usr/lbin/sw/config_clean

定义代理为执行特定于发行版的配置清除步骤而调用的脚本。

适用于 **swagent** 。

请注意：由于 11.31 版和更高版本中不存在转换链接，因此就没有要执行的配置清除步骤，从而决不会对这些版本执行 config_cleanup_cmd。

control_files=

添加或删除控制文件对象时，此选项将列出这些控制文件的标记。没有提供缺省值。如果有多个标记，则必须使用空格来分隔这些标记而且用引号将它们引起来。

适用于 **swmodify** 。

controller_source=

指定控制器为解析选择而访问的软件仓库的位置。设置此选项可以减少控制器与目标之间的网络流量。使用以下目标选择语法来指定位置：

[*host*][**:**][*path*]

该选项对目标使用的源没有任何影响，且在用于交互式用户界面时将被忽略。

适用于 **swcopy** 、 **swconfig** 、 **swinstall** 、 **swremove** 和 **swverify** 。

create_target_acls=true

如果创建目标软件仓库， **swpackage** 将为该软件仓库（如果是新的软件仓库）以及打包到其中的所有产品创建访问控制列表 (ACL)。如果设置为 **false** 且用户为超级用户， **swpackage** 将不会创建 ACL（在将软件打包到分发磁带时， **swpackage** 命令从不创建 ACL）。

适用于 **swpackage** 。

create_target_path=true

可使代理创建目标目录（如果目标目录尚未存在）。如果设置为 **false** ，则不会创建新的目标目录。该选项可防止错误地创建新的目标软件仓库。

适用于 **swcopy** 和 **swinstall** 。

create_time_filter=0

对于汇总源软件仓库，此选项允许 **swlist** 、 **swcopy** 和 **swinstall** 随时间的推移而进行一致的软件选择。如果使用缺省值零，则将根据软件选择和其他选项，包含源软件仓库中所有软件包、产品、子产品和文件集以作为选择候选项（并自动选择相关内容和修补软件）。如果设置为时间（指定为自历元开始的秒数），则只有 create_time 小于或等于该值的软件包、产品和文件集（以及产品中的子产品）可供选择（或自动选择）。要列出软件包、产品和文件集的 create_time，请使用：

swlist -a create_time -a create_date

适用于 **swlist** 、 **swcopy** 和 **swinstall** 。

customer_id=

此编号印刷在软件证书上，用于“解锁”受保护的软件并将安装限制于特定的站点或所有者。可以使用 **-x** *customer_id=* 选项输入编号，也可以使用交互式用户界面输入编号。可以在任何 HP-UX 10.X 或更高版本的系统上使用 *customer_id* 。

适用于 **swinstall** 、 **swcopy** 、 **swlist** 。

defer_configure=false

可使 **swinstall** 在安装运行脚本后，针对 *software_selections* 自动运行这些脚本（未配置备用根目录）。

如果设置为 true，则 **swinstall** 不会运行配置脚本。如果要在以后配置软件，则必须运行 **swconfig** 命令。

注释：

- 如果已配置了其他版本，则不会自动配置一个产品的多个版本。使用 **swconfig** 命令可以分别配置多个版本。
- 如果 SD 安装的软件导致系统重新引导，则将忽略此选项。

适用于 **swinstall** 。

distribution_source_directory=/var/spool/sw

定义源软件仓库的缺省位置（当 **source_type** 为 *directory* 时）。还可以使用 *host*:*path* 语法。**-s** 选项可覆盖该缺省值。

适用于 **swcopy** 、 **swinstall** 和 **swpackage** 。

distribution_target_directory=/var/spool/sw

定义目标软件仓库的缺省分发目录。 *target_selection* 操作数将覆盖此缺省值。

适用于 **swacl** 、 **swcopy** 、 **swlist** 、 **swmodify** 、 **swpackage swreg** 、 **swremove** 和 **swverify**。

distribution_target_serial=/dev/rmt/0m

定义目标磁带设备文件的缺省位置。 *target_selection* 操作数将覆盖此缺省值。

适用于 **swpackage** 。

enforce_dependencies=true

要求在指定的源中或在 *target_selections* 本身解析由 *software_selections* 指定的所有相关关系。

只有相关性已被选定或已按正确的状态（已安装、已配置或可用）存在于目标中时， **swconfig** 、 **swcopy** 和 **swinstall** 命令才会继续执行。这将防止在系统上安装不可用软件。还将确保软件仓库包含可用的软件集。

对于 **swremove** ，如果选定文件集具有相关项（即，其他软件依赖于该文件集）且未选择这些相关项，则不要删除选定的文件集。

如果设置为 **false** ，则将检查相关性，但不会强制执行。共存相关性（如果未强制执行）可能会使选定的软件无法正常工作。必备条件或互斥条件相关性（如果未强制执行）可能会导致安装或配置失败。

适用于 **swconfig** 、 **swcopy** 、 **swinstall** 、 **swremove** 和 **swverify** 。

enforce_dsa=true

在所需的磁盘空间超过受影响的文件系统的可用空间时，防止命令通过分析阶段继续执行。如果设置为 **false** ，则安装、复制或打包操作将使用文件系统的 minfree 空间，而且可能会因到达文件系统的绝对限制而失败。

适用于 **swcopy** 、 **swinstall** 和 **swpackage** 。

enforce_locatable=true

如果设置为缺省值 **true** ，则在命令对无法重定位的文件集尝试重定位时，此选项将生成错误（可重定位的文件集是通过将 **is_relocatable** 属性设置为 **true** 来进行打包的）。如果设置为 **false** ，将覆盖通常的错误处理进程，且 SD 允许命令重定位文件集。

请注意，尽管该选项是为 **swverify** 定义的，但是没有与该选项关联的行为。

适用于 **swinstall** 和 **swverify** 。

enforce_kernbld_failure=true

缺省值 true 可防止在内核构建过程失败时 **swinstall** 命令通过内核构建阶段继续执行。如果设置为 **false** ，则不管系统准备过程或内核构建过程中出现故障还是警告，安装操作都会继续进行。

适用于 **swinstall** 。

enforce_scripts=true

控制对脚本生成的错误进行的处理。如果为 **true** 且脚本返回错误，则该命令将暂停，且屏幕上将出现错误消息，报告执行失败。如果为 **false** ，则脚本生成的错误将被视为警告，且该命令会尝试继续执行。此时屏幕上将出现警告消息并报告该命令已成功。根据需要，该消息将标识出现错误的阶段（配置/取消配置、安装前/安装后、删除前/删除后等）。

适用于 **swask** 、 **swconfig** 、 **swinstall** 和 **swremove** 。

files=

添加或删除文件对象时，此选项将列出这些文件对象的路径名。没有提供缺省值。如果有多个路径名，则这些路径名之间必须使用空格进行分隔。

适用于 **swmodify** 。

fix_explicit_directories=false

控制 **swinstall** 对显式打包的软件（使用显式文件规范打包的软件）所做的响应。缺省值 **false** 可使 **swinstall** 对新目录设置权限（按照产品规范文件中指定的方式），但决不会对原有的目录设置权限。如果设置为 **true**，则 **swinstall** 还会对原有的目录设置权限。

适用于 **swinstall** 。

follow_symlinks=false

请不要按照程序包源文件中的符号链接操作，而应将符号链接包含在打包的产品中。如果此关键字的值为 **true** ，则可使 **swpackage** 按照源文件中的符号链接操作，并将它们所引用的文件包含在打包的产品中。

适用于 **swpackage** 。

force_single_target=false

此选项仅适用于 *HP-UX 10.X*。

此选项仅适用于作为无盘服务器的系统上的交互式用户界面。该选项可使 **swremove** 在单目标模式下运行，即使无盘服务器通常使得 **swremove** 在多目标模式下运行也是如此。

适用于 **swremove** 。

include_file_revisions=false

请不要将每个源文件的修订版属性包含在已打包的产品中。由于此操作非常耗时，因此缺省情况下，不会包括修订版属性。如果设置为 **true** ， **swpackage** 将执行 *what*(1) 并可能执行 *ident*(1) （按该顺序）以尝试确定文件的修订版属性。

适用于 **swpackage** 。

install_cleanup_cmd=/usr/lbin/sw/install_clean

定义代理为了在最后一个后期安装脚本运行后立即执行特定于发行版的安装清除步骤，而调用的脚本。对于操作系统更新，此脚本至少应删除已通过 **install_setup** 脚本保存的命令。该脚本是在安装了所有的文件集后且在重新引导至新的操作系统之前执行的。

适用于 **swagent** 。

请注意：由于 11.31 版和更高版本中不存在转换链接，因此就没有要执行的安装清除步骤，从而从不对这些版本执行 install_cleanup_cmd。

installed_software_catalog=products

定义存储已安装产品数据库 (IPD) 的目录路径。此信息对已安装的软件进行了说明。当设置为绝对路径时，此选项定义 IPD 的位置。当此选项包含相对路径时， SD 控制器将该值追加到 **admin_directory** 选项指定的值以确定 IPD 的路径。对于备用根目录，将相对于备用根目录的位置解析该路径。此选项不影响安装软件的位置，只影响 IPD 位置。

此选项允许多个用户或多个进程同时安装和删除多个软件应用程序，每个应用程序或应用程序组使用不同的 IPD。

注意：使用特定的 **installed_software_catalog** 管理特定的应用程序。SD 不支持同一个应用程序在多个 IPD 中有多个说明。

另请参阅 **admin_directory** 和 **run_as_superuser** 选项，它们控制 SD 的非特权模式（此模式仅用于管理经过特殊设计和打包的应用程序。此模式不能用于管理 HP-UX 操作系统或其修补软

件。有关非特权 SD 的完整说明，请参阅 **http://docs.hp.com/zh_cn/** 上的《Software Distributor 管理指南》）。

适用于除 **swacl** 、 **swask** 、 **swconfig** 、 **swinstall** 、 **swlist** 、 **swmodify** 、 **swremove** 和 **swverify** 以外的所有 SD 命令。

install_setup_cmd=/usr/lbin/sw/install_setup

定义代理为执行特定于发行版的安装清除步骤而调用的脚本。对于操作系统更新，此脚本至少应该将检查安装、安装前和安装后脚本所需的命令复制到一条路径，以便在更新实际命令的同时可以在该路径中访问这些命令。在加载内核文件集之前将执行此脚本。

适用于 **swagent** 。

请注意：由于 11.31 版和更高版本中不存在转换链接，因此就没有要执行的安装设置步骤，从而从不对这些版本执行 install_setup_cmd。

job_title= 这是为作业提供标题的 ASCII 字符串。调用 **swjob** 时，该字符串将与作业 ID 一同显示，以提供有关作业的附加标识信息。

适用于 **swconfig** 、 **swcopy** 、 **swinstall** 、 **swremove** 和 **swverify** 。

kernel_build_cmd=/usr/sbin/mk_kernel

定义代理为构建内核而调用的脚本。

适用于 **swagent** 。

kernel_path=/stand/vmunix

定义系统的可引导内核的路径。此路径会通过 **SW_KERNEL_PATH** 环境变量传递至 **kernel_build_cmd** 。

适用于 **swagent** 。

layout_version=1.0

指定在写入分发以及 **swlist** 输出时，SD 命令遵循的 POSIX **layout_version** 。支持的值为“1.0”（缺省值）和“0.8”。HP-UX 10.10 版及其更高版本的 SD 可以读取或写入这两个布局版本中的任意一个版本。

SD 对象和属性语法遵循《IEEE POSIX 1387.2 Software Administration》标准的 **layout_version 1.0** 规范。SD 命令仍接受与早期的布局版本相关联的关键字名称，但只能使用 **layout_version=0.8** 来创建可由 SD 的早期版本读取的分发。

在产品规范文件 (PSF) 中指定 *layout_version* 属性可以控制 **swpackage** 使用的版本。然而，如果 *layout_version* 在 PSF 中的属性为 1.0，则 *is_locatable* 属性在所有情况下都缺省为 true，因此必须显式设置为 false （有关 PSF 的详细信息，请参阅 *swpackage*(4) ）。

布局版本 1.0 增加了那些仅支持 0.8 版的系统无法识别的大量功能，其中包括：

- Category 类对象（以前是 bundle 或 product 类内部的 **category** 和 **category_title** 属性）。
- 修补软件处理属性，包括 **applied_patches**、**is_patch** 和 **patch_state** 。
- 文件集 **architecture** 属性，允许指定产品在其中运行的目标系统的体系结构。

除增加新的属性和对象之外，layout_version 1.0 对下列已有的 0.8 版的对象和属性作了如下更改：

- 用具有 **sequence_number** 属性的 **media** 对象替换软件仓库的 **media_sequence_number** 。
- 用 **vendor_tag** 属性和在产品或软件包外部定义的相应 **vendor** 对象替换产品和软件包内部的 **vendor** 定义。
- 将 **corequisite** 和 **prerequisite** 文件集属性用复数形式表示（表示为 **corequisites** 和 **prerequisites** ）。
- 将 **timestamp** 属性更改为 **mod_time** 。

适用于 **swpackage** 、 **swcopy** 、 **swmodify** 和 **swlist** 。

level= 指定 **swlist** 、 **swacl** 或 **swreg** 的软件 *level* 。

对于 **swlist** ：

列出指定 *level* 下的所有对象。指定级别和指定 *software_selections* 的深度均可以控制 **swlist** 输出的深度。支持的软件级别包括：

bundle 显示软件包级别下的所有对象。
product 显示产品级别下的所有对象。还可以使用 *-l bundle -l product* 显示软件包。
subproduct 显示子产品级别下的所有对象。
fileset 显示文件集级别下的所有对象。还可以使用 *-l fileset -l subproduct* 显示子产品和文件集。
file 显示文件级别下的所有对象（即软件仓库、产品、文件集以及文件）。
control_file 显示 control_file 级别下的所有对象。
category 显示所有类别的可用软件对象。
patch 显示所有已应用的修补软件。

支持的软件仓库和根目录级别包括：

depot 仅显示软件仓库级别（即位于指定目标主机的软件仓库）。
root 列出所有备用根目录。
shroot 列出所有已注册的共享根目录 （仅限 *HP-UX 10.X*）。
prroot 列出所有已注册的专用根目录 （仅限 *HP-UX 10.X*）。

对于 **swacl** ：

level 选项定义了要查看或修改的 ACL 的级别：

host 查看（或修改）ACL 保护的由 *target_selections* 标识的主机系统。

depot 查看（或修改）ACL 保护的由 *target_selections* 标识的软件仓库。

root 查看（或修改）保护 *target_selections* 所标识的根文件系统的 ACL。

product 查看（或修改）ACL 保护的由 *software_selection* 标识的软件产品。仅适用于软件仓库中的产品，不适用于根目录中已安装的产品。

product_template
查看（或修改）模板 ACL，该 ACL 用于初始化以后添加到软件仓库（由 *target_selections* 标识）的产品的 ACL。

global_soc_template
查看（或修改）ACL，该 ACL 用于初始化以后添加到主机（由 *target_selections* 标识）的软件仓库或根文件系统的 ACL。

global_product_template
查看（或修改）ACL，该 ACL 用于初始化以后添加到主机（由 *target_selections* 标识）的软件仓库的 **product_template** 。

对于 **swreg** ：

level 选项定义要注册或取消注册的对象的级别：

depot 指定目标主机上的软件仓库。

root 所有备用根目录。

shroot 所有已注册的共享根目录 （仅限 *HP-UX 10.X*）。

prroot 所有已注册的专用根目录 （仅限 *HP-UX 10.X*）。

适用于 **swacl** 、 **swlist** 和 **swreg** 。

log_msgid=0

将数字标识号添加到 SD 日志文件消息的开头：

0 （缺省值）消息中不附加标识符。

1 仅为错误消息添加标识符。

2 为错误消息和警告消息添加标识符。

3 为错误消息、警告消息以及说明消息添加标识符。

4 为错误消息、警告消息、说明消息以及其他特定信息性消息添加标识符。

适用于 **swconfig** 、 **swcopy** 、 **swinstall** 、 **swmodify** 、 **swpackage** 、 **swreg** 、 **swremove** 和 **swverify** 。

logdetail=false

logdetail 选项控制写入日志文件的详细信息数量。如果设置为 **true** ，此选项会将详细的任务信息（如指定的选项、进度说明以及附加的摘要信息）添加到日志文件。此信息是 **loglevel** 选项控制的日志信息的补充。

下面是 **loglevel** 和 **logdetail** 选项的可能组合：

日志级别	日志详细信息	包含的信息
loglevel=0		不向日志文件写入信息。
loglevel=1	***logdetail=false***	仅记录关键事件；这是缺省设置。
loglevel=1	***logdetail=true***	记录以上事件详细信息，以及任务进度消息。 loglevel=1 不是必需的设置，只是缺省设置。
loglevel=2	***logdetail=false***	仅记录事件和文件级消息。 logdetail=false 选项不是必需的设置。
loglevel=2	***logdetail=true***	记录所有信息。必须设置 loglevel=2 和 logdetail=true 选项。该组合可能产生与以前的 HP-UX 10.x 发行版相同的日志文件行为。

适用于 **swconfig** 、 **swcopy** 、 **swinstall** 、 **swreg** 、 **swremove** 和 **swverify** 。

logfile=/var/adm/sw/sw<command>.log

定义每个 SD 命令的缺省日志文件（代理日志文件的位置始终相对于目标软件仓库或目标根文件夹，例如， **/var/spool/sw/swagent.log** 和 **/var/adm/sw/swagent.log** ）。

适用于除 **swacl** 、 **swlist** 和 **swjob** 之外的所有命令。

loglevel=1 通过在 SD 日志文件消息前添加标识号，控制记录到命令日志文件、目标代理日志文件以及源代理日志文件的事件的日志级别。此信息是 **logdetail** 选项控制的详细信息的补充。有关详细信息，请参阅 **logdetail** 。

值：

0 不向日志文件提供信息。

1 启用详细日志记录，以记录到日志文件。

2 启用非常详细的日志记录，以记录到日志文件。

适用于 **swconfig** 、 **swcopy** 、 **swinstall** 、 **swmodify** 、 **swpackage** 、 **swremove** 和 **swverify**。

match_target=false

如果设置为 **true** ，则通过定位与目标系统上已安装文件集匹配的源系统上的文件集来进行软件选择。如果指定了多个目标，则使用列表中的第一个目标作为选择的基础。

适用于 **swinstall** 。

max_agents=-1

允许同时运行的代理的最大数量。值 -1 表示没有限制。

适用于 **swagentd** 。

max_targets=25

如果设置为正整数，则此选项将并发安装或复制操作的数量限制为指定数量。完成每个复制或安装操作时，直到完成所有目标之后才会选择并启动另一个目标。

服务器和网络性能决定最佳设置；建议起点为 25（缺省值）。如果将此选项设置为小于 1 的值，则 SD 尝试一次性安装或复制到所有目标。

适用于 **swcopy** 和 **swinstall** 。

media_capacity=1330

如果创建分发磁带或多目录介质（如 CD-ROM），则此关键字以百万字节为单位（而不是以兆字节为单位）指定磁带的容量。如果介质不是 DDS 磁带或磁盘文件，则该选项为必需选项。如果未设置该选项，则 **swpackage** 将大小设置为缺省值 1,330 兆字节（针对磁带），或设置为最大值为 **minfree** 的可用磁盘空间数量（针对磁盘文件）。如同对多序列介质一样，SD 在多目录介质间也使用相同的格式，包括基于文件集分区和 **media_sequence_number** 属性设置计算正确大小。

适用于 **swpackage** 。

media_type=directory

定义要创建的分发类型。可识别的类型为 **directory** 和 **tape** 。

适用于 **swpackage** 。

minimum_job_polling_interval=1

定义唤醒守护程序以扫描作业队列中必须启动的调度作业的频率（以分钟为单位)。如果设置为 0，则没有启动的调度作业。

适用于 **swagentd** 。

mount_all_filesystems=true

缺省情况下，SD 命令在分析阶段之初就会尝试挂接 **/etc/fstab** 文件中的所有文件系统，以确保在继续操作之前所有列出的文件系统都得到挂接。该策略有助于确保文件不会被加载到未来挂接点之下的目录中，并且确保预期文件可用于删除或验证操作。

如果设置为 **false** ，则不会尝试挂接操作，并且不会检查当前挂接。

适用于 **swconfig** 、 **swcopy** 、 **swinstall** 、 **swremove** 和 **swverify** 。

mount_cmd=/sbin/mount

定义代理挂接所有文件系统而调用的命令。

适用于 **swagent** 。

objects_to_register=

定义要注册或取消注册的缺省对象。没有提供缺省值（请参阅上文中的 **select_local** ）。如果有多个对象，则必须用空格进行分隔。

适用于 **swreg** 。

one_liner=

定义在非详细列表中列出的属性。

适用于 **swlist** 和 **swjob** 。

os_name 该选项可以与 **os_release** 结合使用，以为 HP-UX 更新指定文件集选择。只能从命令行指定 **os_name** 。有关正确的语法，请参考 SD 的 **README** 文件。可以通过输入以下命令显示 **README** 文件：

swlist -d -a readme SW-DIST [@ *host:/depot* **]**

适用于 **swinstall** 。

os_release 该选项可以与 **os_name** 结合使用，以为 HP-UX 更新指定文件集选择。只能从命令行指定 **os_release** 。有关正确的语法，请参考 SD 的 **README** 文件。可以通过输入以下命令显示 **README** 文件：

swlist -d -a readme SW-DIST [@ *host:/depot* **]**

适用于 **swinstall** 。

package_in_place=false

如果设置为 **true** ， **swpackage** 将对指定产品进行打包，这样目标软件仓库将不包含组成产品的文件。相反， **swpackage** 将插入对用于构建产品的原始源文件的引用。这样无需消耗将所有源文件复制到目标软件仓库所需的整个磁盘空间，就可实现产品打包。

适用于 **swpackage** 。

patch_commit=false

通过删除为修补软件回滚而保存的文件来提交修补软件。如果设置为 **true** 并与 **swmodify** 一起运行，则只有删除修补软件更改的关联基本软件，才能回滚（删除）修补软件。

适用于 **swmodify** 。

patch_filter=*.*

指定修补软件过滤器的 *software_specification* 。缺省值为 ***.*** 。

该选项可与 **autoselect_patches** 和 **patch_match_target** 选项结合使用，用于过滤符合 *software_specification* 指定条件的选定修补软件。

适用于 **swask** 、 **swcopy** 和 **swinstall** 。

patch_match_target=false

如果设置为 **true** ，则该选项将选择与目标根目录或软件仓库上的软件相对应的最新修补软件（由 *is_patch* 属性标识的软件）。

选项 **patch_filter=** 可与 **patch_match_target** 结合使用。

适用于 **swcopy** 和 **swinstall** 。

patch_one_liner=title patch_state

指定在调用了 **-l patch** 选项但未指定 **-a** 或 **-v** 选项时，显示的每个列出对象的属性。缺省显示属性为 **title** 和 **patch_state** 。

适用于 **swlist** 。

patch_save_files=true

保存修补的文件，以允许将来回滚修补软件。如果设置为 **false** ，则只有同时删除修补软件修改的基本软件，才能回滚（删除）修补软件。

适用于 **swinstall** 。

polling_interval=2

定义交互式 (GUI) 会话使用的轮询间隔（以秒为单位）。它指定每个目标代理多长时间将被轮询一次，以获得正在执行的任务的状态信息。在广域网之间操作时，可以增加轮询间隔以减少网络开销。

适用于 **swcopy** 、 **swinstall** 和 **swremove** 。

preserve_create_time=false

复制软件仓库时保留原始创建时间，以便使用这些副本时能够产生一致的结果。如果设置为缺省值 **false** ，则软件包、产品以及文件集的 create_time 将设置为与在软件仓库中创建对象的时间相等。如果设置为 **true** ，软件包、产品以及文件集的 create_time 设置为源软件仓库中指定的时间。请注意，当向主软件仓库复制时，使用该选项可以更改那些在使用 **create_time_filter** 选项时可见的对象。

适用于 **swcopy** 。

reboot_cmd=/sbin/reboot

定义代理重新引导系统时调用的命令。

适用于 **swagent** 。

reconfigure=false

防止重新配置已处于已配置状态的软件。如果设置为 **true** ，则可以重新配置已配置的软件。

适用于 **swconfig** 。

recopy=false

该选项防止 SD 重新复制（覆盖）文件集的现有版本。如果设置为 **true** ，则将重新复制文件集。

适用于 **swcopy** 。

register_new_depot=true

可使 **swcopy** 用本地 **swagentd** 注册一个新创建的软件仓库。该操作允许其他 SD 命令自动“查看”该软件仓库。如果设置为 **false** ，则不会自动注册一个新的软件仓库（可以在以后用 **swreg** 命令注册）。

适用于 **swcopy** 。

register_new_root=true

可使 **swinstall** 用本地 **swagentd** 注册一个新创建的备用根目录。该操作允许其他 SD 命令自动“查看”该根目录。如果设置为 **false** ，则不会自动注册一个新的根目录（可以在以后用 **swreg** 命令注册）。

适用于 **swinstall** 。

reinstall=false

该选项防止 SD 重新安装（覆盖）文件集的现有修订版。如果设置为 **true** ，则将覆盖文件集。

适用于 **swinstall** 。

reinstall_files=false

控制文件的覆盖，这可能提高低速网络或磁盘的性能。设置为缺省值 false 时， SD 将源文件集中的每个文件与目标系统上的对应文件进行比较。SD 基于大小、时间戳以及 *checksum*（可选）对文件进行比较（请参阅 **reinstall_files_use_cksum** ）。如果文件相同，则不会覆盖目标系统上的文件。

设置为 true 时，SD 将不比较文件而覆盖目标上的所有相同文件。

适用于 **swinstall** 、 **swcopy** 和 **swpackage** 。

reinstall_files_use_cksum=true

（对于 **swpackage** ，该选项的缺省值为 **false** ）。当 **reinstall_files** 选项设置为 false 时，控制校验和比较的使用。设置为缺省值 true 时，该选项使 SD 计算并比较校验和，以确定新文件是否应当覆盖旧文件。校验和的使用降低了比较速度，但在相等性检查方面要比大小和时间戳更强大。

如果设置为 false，SD 不计算校验和而只比较文件的大小和时间戳。

适用于 **swcopy** 、 **swinstall** 和 **swpackage** 。

remove_empty_depot=true

在删除最后一个产品时删除空的软件仓库。如果设置为 **false** ，则不会删除空的软件仓库，并保留所有软件仓库 ACL。

适用于 **swremove** 。

remove_obsolete_filesets=false

控制 **swcopy** 是否自动从目标软件仓库中的目标产品中删除过时的文件集。如果设置为 **true**，**swcopy** 将删除在复制过程中写入目标产品的过时文件集。删除操作在复制操作完成后进行。如果文件集不是位于源软件仓库上的产品的最新打包部分，则这些文件集被定义为过时。

适用于 **swcopy** 。

remove_setup_cmd=/usr/lbin/sw/remove_setup

定义代理为执行特定于发行版的删除准备而调用的脚本。对于操作系统更新，如果删除了文件集，则该脚本将调用 **tlink** 命令。

适用于 **swagentd** 。

请注意：由于 11.31 版和更高版本中不存在转换链接，因此就没有要执行的删除准备步骤，从而从不对这些版本执行 remove_setup_cmd。

retry_rpc=1

定义在文件传输过程中，断开（超时）的源连接的重试次数。与 **rpc_timeout** 选项结合使用时，可以增加通过低速或繁忙网络成功安装的可能性。如果设置为 0，任何针对源主机的 **rpc_timeout** 会导致任务异常中止。如果设置为从 1 到 9 的值，则将按照该值指定的次数尝试安装每个文件集（可以使用 **retry_rpc_interval** 选项指定每次重试尝试之间的间隔长度）。

选项 **reinstall_files** 也应设置为 false，以免在已成功安装的文件集内安装文件。

该选项也适用于与代理联系的控制器。如果由于某种原因导致代理会话启动失败，控制器尝试按 **retry_rpc** 指定的次数重新连接代理，并使用 **retry_rpc_interval** 选项的值决定重新连接代理的每次尝试之间需要等待多长时间。

适用于 **swcopy** 和 **swinstall** 。

retry_rpc_interval={0}

指定初始连接失败之后重复尝试连接目标主机的间隔长度（以分钟为单位）。与 **retry_rpc** 选项结合使用。如果该选项中的数值等于 **retry_rpc** 选项的值，SD 将尝试按照 **retry_rpc** 中指定的次数重建源连接。如果 **retry_rpc_interval** 中的数值小于 **retry_rpc** 的值，则 SD 将重复最后间隔值，直到重试次数与 **retry_rpc** 相符。

比如，如果代理会话启动失败，并且 **retry_rpc** 设置为 9，同时 **retry_rpc_interval** 设置为 {1 2 4 8 15} 以允许长时间等待来处理暂时网络失败，则 SD 控制器将在 1 分钟后尝试第一次重新联系代理，2 分钟后尝试第二次，4 分钟后尝试第三次，然后 8 分钟后、 15 分钟后尝试所有剩余的重试，直到 9 次重试完毕。对于这些值，文件加载失败可能导致操作暂停 90 分钟 (1+2+4+8+15+15+15+15+15)。如果 **retry_rpc** 设置为 5，且 **retry_rpc_interval** 设置为 {1 2 4 8 15}，则控制器将在 30 分钟的时间内尝试连接目标五次。

适用于 **swcopy** 和 **swinstall** 。

rpc_binding_info=ncacn_ip_tcp:[2121] ncadg_ip_udp:[2121]

定义守护程序可依此进行监听且其他命令可用来联系该守护程序的协议序列和端点。如果一个协议序列的连接失败，则将尝试下一个协议序列。在大多数平台上，SD 支持 **tcp** (**ncacn_ip_tcp:[2121]**) 和 **udp** (**ncadg_ip_udp:[2121]**) 协议序列。

值（或 **swagentd** 的值）可以为下列格式：

- 包含一个协议序列和一个端点的 DCE 字符串绑定。语法为： **protocol_sequence:[endpoint]** 。
- 没有指定端点的 DCE 协议序列的名称。语法为： **protocol_sequence** ，例如 **ncadg_ip_udp** 或 **ncacn_ip_tcp** （结尾 **:** 可附加到协议序列后面，它没有任何影响）。由于没有指定端点，因此必须运行 DCE 端点映射程序 *rpcd* ，并且此端点映射程序将用于查找由 **swagentd** 注册的端点。
- 文字字符串 **all** 。此条目意味着使用（尝试）DCE RPC 支持的所有协议序列。它应当是此列表中的唯一条目。要使用此选项，还必须运行 DCE 端点映射程序 **rpcd** 。

适用于除 **swask** 、 **swpackage** 和 **swmodify** 之外的所有命令。

rpc_binding_info_alt_source=ncadg_ip_udp:[2121]

定义代理尝试联系 **alternate_source** 选项指定的备用源软件仓库时使用的协议序列和端点。SD 支持 **udp** (**ncadg_ip_udp:[2121]**) 和 **tcp** (**ncacn_ip_tcp:[2121]**) 协议序列（或端点）。

适用于 **swagent** 。

rpc_binding_info_source=

定义用于联系进行源访问的守护程序的协议序列和端点。如果未设置任何值（缺省），将使用 **rpc_binding_info** 的值。

适用于 **swinstall** 和 **swcopy** 。

rpc_binding_info_target=

定义用于联系进行目标访问的守护程序的协议序列和端点。如果未设置任何值（缺省值），将使用值 **rpc_binding_info** 。

适用于 **swinstall** 和 **swcopy** 。

rpc_timeout=5

通信超时的相对长短。此值是一个 0 到 9 之间的值，由 DCE RPC 解释。较高的值表示时间较长；对于速度慢或繁忙的网络，可能需要使用较高的值。较低的值将更快识别对未启动的主机或未运行 **swagentd**. 的主机的通信尝试。每个值大约是前一个值的两倍。对于 **ncadg_ip_udp** 协议序列，值为 5 时大约是 30 秒。当使用 **ncacn_ip_tcp** 协议序列时，此选项可能没有任何明显的影响。

适用于除 **swpackage** 和 **swmodify** 以外的所有命令。

run_as_superuser=true

此选项控制 SD 的非特权模式。当调用用户是超级用户时，将忽略此选项（视为 true）。

当设置为缺省值 true 时，将以正常方式执行 SD 操作，此时的操作权限是授予本机超级用户的或由 SD ACL 设置（有关 ACL 的详细信息，请参阅 *swacl*(1M)）。

当设置为 false，并且调用用户是本机用户但 不是超级用户时，将调用非特权模式：

- 此时的操作权限基于用户的文件系统的权限。
- 忽略 SD ACL。
- SD 创建的文件具有调用用户的 UID 和 GID，所创建文件的模式是根据调用用户的 umask 设置的。

SD 的非特权模式仅用于管理经过特殊设计和打包的应用程序。此模式不能用于管理 HP-UX 操作系统或其修补软件。有关非特权 SD 的完整说明，请参阅《Software Distributor 管理指南》（位于 **http://docs.hp.com/zh_cn/** 网站）。

另请参阅 **admin_directory** 和 **installed_software_catalog** 选项。

适用于除 **swagent**、 **swagentd** 和 **install-sd** 以外的所有 SD 命令。

show_superseded_patches=false

显示或隐藏 **swlist** 输出中的替代修补软件。在缺省状态 **false** 下，即使对替代修补软件执行 **swlist** 命令，" "swlist" 也不会显示替代修补软件。如果将此选项设置为 **true**，则允许显示替代修补软件。

适用于 **swlist**。

select_local=true

如果未指定 *target_selections*，则选择本地主机的缺省 **target_directory** 作为命令的 *target_selection*。

适用于 **swacl**、 **swconfig**、 **swcopy**、 **swinstall**、 **swlist**、 **swreg**、 **swremove** 和 **swverify**。

software= 定义缺省 *software_selections*。没有提供缺省值。如果有多个软件选择，则必须用空格进行分隔。软件通常在软件输入文件中指定为命令行或 GUI 中的操作数。

适用于除 **swreg** 和 **swjob** 以外的所有命令。

software_view=products

指示命令的交互式界面使用的软件视图以及 **swlist** 用于显示缺省列表级别的软件视图。可以将其设置为 products、all_bundles 或软件包类别标记（以指明仅显示该类别的软件包）。

适用于 **swcopy**、 **swinstall**、 **swlist** 和 **swremove**。

source= 指定自动绕过 GUI 和 CLI 源选择对话框的源。它与 **-s***source* 命令行选项的作用相同。使用以下语法指定源。

[*path*]

适用于 **swcopy** 和 **swinstall** 。

source_cdrom=/SD_CDROM

使用以下语法定义源 CD-ROM 的缺省位置 [*host*]**:**[*path*] 。

适用于 **swinstall** 。

source_depot_audit=true

如果源计算机和目标计算机已升级到 SD B.11.00 版或更高版本，则源软件仓库计算机上的系统管理员可以设置此选项，以跟踪 哪些用户从源计算机上的软件仓库上取出 哪个软件，以及何时取出软件（请注意，在目标计算机上运行 **swinstall/swcopy** 的用户无法设置此选项；只有源软件仓库计算机的管理员才能设置此选项）。

source_depot_audit 设置为 **true** 时，将在源软件仓库上创建 **swaudit.log** 文件（针对可写入的目录软件仓库）或在 **/var/tmp** 下创建该文件（针对 *tar* 图像、CD-ROM 或其他不可写入的软件仓库）。

用户可以调用 **swlist** 交互式用户界面（使用 **swlist -i -d** ）来查看、打印或保存远程或本地软件仓库上的审核信息。用户可以基于语言优先级查看审核信息，但是系统必须具有相应的 SD 消息清单文件。例如，用户可以在 **swlist** 的某次调用过程中查看用日语显示的源审核信息，然后在下次调用过程中查看用英语显示的相同信息。

适用于 **swagent** 。

source_file=psf

定义源产品规范文件 (PSF) 的缺省位置。不允许使用 *host*:*path* 语法，只能指定有效的 **path**。**-s** 选项可覆盖该值。

适用于 **swpackage** 和 **swmodify** 。

source_tape=/dev/rmt/0m

定义源磁带的缺省位置，通常是本地磁带设备的字符专用文件。还可以使用 *host*:*path* 语法，但是 host 必须匹配本地主机。 **-s** 选项可覆盖该值（请注意，SD 可以读取 **tar** 和 **cpio** 磁带软件仓库）。

适用于 **swcopy** 和 **swinstall** 。

source_type=directory

定义缺省源类型： **cdrom** 、 **file** 、 **directory** 或 **tape** 。从 **-s** 选项派生的源类型将覆盖该值（请注意，SD 可以读取 **tar** 和 **cpio** 磁带软件仓库）。

适用于 **swcopy** 、 **swinstall** 和 **swpackage** （值 **cdrom** 和 **tape** 仅适用于 **swcopy** 和 **swinstall**。值 **file** 仅适用于 **swpackage** ）。

system_file_path=/stand/system

定义内核模板文件的路径。此路径会通过 **SW_SYSTEM_FILE_PATH** 环境变量传递至 **system_prep_cmd** 。

适用于 **swagent** 。

system_prep_cmd=/usr/lbin/sysadm/system_prep

定义代理调用的内核构建准备脚本。此脚本必须进行必要的准备，这样控制脚本才能正确配置要构建的内核。在所有内核文件集加载完成前调用此脚本。

适用于 **swagent** 。

targets= 定义缺省 *target_selections* 。没有提供缺省值（请参阅上文中的 **select_local** ）。如果有多个目标选择，则必须用空格进行分隔。目标通常在目标输入文件中指定为命令行或 GUI 中的操作数。

适用于所有命令。

uncompress_cmd=

定义用于在安装、复制或打包时解压缩文件的命令。此命令将处理以压缩格式保存在介质上的文件。如果文件的 **compression_type** 为 **gzip** ，则使用内部解压缩 (**funzip**) ，而不使用外部解压缩 **uncompress_cmd** 。

适用于 **swpackage** 和 **swagent** 。

uncompress_files=false

如果从源传输的文件已被压缩，则设置此选项将在文件存储到目标软件仓库前对其进行解压缩。

适用于 **swcopy** 和 **swpackage** 。

use_alternate_source=false

使每个目标代理使用自身的配置替代源，而不使用用户指定的替代源。如果设置为 **false** ，则每个目标代理将使用相同的源，即由用户指定并由此命令验证的源。如果设置为 **true** ，则每个目标代理将使用自身的源配置值。

适用于 **swcopy** 和 **swinstall** 。

verbose= 控制非交互式命令输出的详细程度：

0 禁止输出到 stdout（始终将错误和警告消息写入 stderr）。

1 允许将详细信息写入 stdout。

2 对于 **swpackage** 和 **swmodify** ，启用非常详细的消息输出，以输出到 stdout。

对于 **swlist** 命令，详细列表包括已经为每个 *software_selection* 操作数的适当级别定义的所有属性。这些属性按每行一个的方式列出，并以属性关键字开头。

如果设置为 0， **-v** 选项将覆盖此缺省值。

适用于所有命令。

write_remote_files=false

防止对远程 (NFS) 文件系统执行文件操作。跳过要在远程 (NFS) 文件系统的目标上安装、复制、删除或封装的所有文件。

如果设置为 true，且超级用户具有远程文件系统的写入权限，则不会跳过远程文件，而是会安装、复制、打包或删除。

适用于 **swconfig** 、 **swcopy** 、 **swinstall** 、 **swpackage** 和 **swremove** 。

会话文件

每次调用 SD 命令都会定义一个 任务会话。大多数 SD 命令会在任务实际开始执行前自动保存任务会话信息（选项、源信息、软件选择以及目标选择）。这样，即使会话在任务完成之前结束，用户也可以重新执行该命令。还可以保存命令行和交互式会话的会话信息。

在命令行中，通过执行带 **-C** *session__file* 选项的此命令可以保存会话信息。可以为会话文件指定相对路径或绝对路径。如果未指定目录，则缺省位置为 **$HOME/.sw/sessions/** 。

在交互式会话中，可以随时通过从 *File* 菜单中选择 *Save Session* 或 *Save Session As* 选项，将会话信息保存到文件。

会话信息保存到以下文件：

$HOME/.sw/sessions/*command_name***.last**

例如：

/home/my_user_name/.sw/sessions/swinstall.last

每次调用该命令时将覆盖此文件。会话文件的内容使用以下语法：

[*command_name*.]*option*=*value*

command_name 前缀表示保存会话信息的 SD 命令名称。例如：

swpackage.verbose=3

要从命令行重新执行会话，请将会话文件指定为 **-S** 选项的参数。

要从交互式会话中重新执行已保存的会话，请使用 *File* 菜单的 *Recall Session* 选项。

重新执行会话文件时，会话文件中的值将优先于系统缺省文件中的值。同样，所有命令行选项和参数优先于会话文件中的值。

软件和目标列表

大多数 SD 命令支持独立输入文件中的软件选择和目标选择（请参阅 **-f** 和 **-t** 命令行选项）。将选择对这些文件中指定的软件和目标进行操作。 **swinstall** 和 **swcopy** 还支持交互式读取和保存目标和软件组。目标和软件组可以保存在文件中（缺省位置 **$HOME/.sw/targets/** 和 **$HOME/.sw/software/** ），然后可以在后续 **swinstall** 和 **swcopy** 操作中进行选择。

此外，支持交互式界面的命令将从以下位置的值中读取要对其执行操作的可能主机的列表：

/var/adm/sw/defaults.hosts 系统级缺省主机列表，

$HOME/.sw/defaults.hosts 用户特定的缺省主机列表。

此文件中的主机没有标记为要运行，但是提供了供选择的缺省列表。对于每个交互式命令，将在单独的列表（分别为 **hosts** 和 **hosts_with_depots** ）中指定包含根目录和软件仓库的目标主机。主机列表括在花括号 {} 中，并由空白字符（空格、制表符和换行符）分隔。例如：

```
swinstall.hosts={hostA hostB hostC hostD
hostE hostF}
swcopy.hosts_with_depots={hostS}
swremove.hosts={hostA hostB hostC hostD
hostE hostF}
swremove.hosts_with_depots={hostS}
```

大多数 SD 命令支持使用 **-x patch_filter=***software_specification* 选项进行修补软件过滤。此外，交互式用户界面命令 **swinstall** 和 **swcopy** 将读取可能的修补软件过滤器的列表。可以使用此列表中的值作为选择条件。这些列表存储在以下位置：

/var/adm/sw/defaults.patchfilters
系统级缺省修补软件过滤器列表。

$HOME/.sw/defaults.patchfilters
用户特定的缺省修补软件过滤器列表。

此文件中的过滤器没有标记为用于选择，但是提供了可供选择的缺省列表。修补软件过滤器列表用 {} 括号括起，并且用空白符（空白、制表符或换行符）分隔。例如：

```
swinstall.patch_filter_choices={
*.*,c=enhancement
*.*,c=critical
}
swremove.patch_filter_choices={
Product.Fileset,c=halts_system
}
```

环境变量

SD 程序受外部环境变量的影响，设置控制脚本使用的环境变量，并且设置影响 **swinstall** 和 **swremove** 运行的脚本的附加环境变量。

影响 SD 命令的外部环境变量:

LANG 确定显示消息的语言。如果未指定 LANG 或将其设置为空字符串，则使用缺省值 **C** 。有关详细信息，请参阅 *lang*(5) 。

注释：用于显示 SD 代理和守护程序日志消息的语言是由系统配置变量脚本 **/etc/rc.config.d/LANG** 设置的。例如，要用日语显示代理和守护程序日志消息，则必须将 **/etc/rc.config.d/LANG** 设置为 **LANG=ja_JP.SJIS** 或 **LANG=ja_JP.eucJP** 。

此变量适用于所有 SD 命令。

LC_ALL 确定语言环境，该语言环境用于覆盖由 **LANG** 的设置或任何以 **LC_** 开头的环境变量指定的语言环境类别的任何值。

LC_CTYPE 确定将文本数据字节序列解释为何种字符（例如供应商定义的属性值中的单字节字符和多字节字符）。

LC_MESSAGES
确定写入消息的语言。

LC_TIME 确定当由 **swlist** 显示时的日期格式（ create_date 和 *mod_date* ）。所有实用程序在 **stdout**、 **stderr** 和 **logging** 中显示日期和时间时使用此格式。

TZ 确定当显示日期和时间时使用的时区。

影响脚本的环境变量:

SW_CATALOG
保存已安装产品数据库 (IPD) 的路径，此路径相对于 **SW_ROOT_DIRECTORY** 环境变量中的路径。请注意，可以使用 **installed_software_catalog** 缺省选项指定 IPD 的路径。

SW_CONTROL_DIRECTORY
定义正在执行的脚本的当前目录，可能是临时清单目录，或者是已安装产品数据库 (IPD) 中的目录。此变量通知脚本软件的其他控制脚本的位置（例如下标）。

SW_CONTROL_TAG
保存正在执行的 *control_file* 的标记名。对软件进行打包时，可以在软件仓库中定义控制文件的实际名称和路径。可使用户为 *control_file* 定义不同于其标记的名称，并且允许用户使用多个控制文件定义指向同一文件。 *control_file* 可以查询 **SW_CONTROL_TAG** 变量来确定正在执行的标记。

SW_LOCATION

定义产品的位置，此位置可能与缺省产品目录不同。与 **SW_ROOT_DIRECTORY** 结合时，此变量通知脚本产品文件的位置。

SW_PATH 定义用于控制脚本的最小命令集的 **PATH** 变量（例如 **/sbin:/usr/bin** ）。

SW_ROOT_DIRECTORY

定义运行会话的根目录，可以为 **/** 或备用根目录。此变量通知控制脚本安装产品的根目录。脚本必须使用此目录作为 **SW_LOCATION** 的前缀，以定位产品的已安装文件。仅当 **SW_ROOT_DIRECTORY** 为 **/** 时才运行配置脚本。

SW_SESSION_OPTIONS

包含文件的路径名，该文件包含特定命令的每个选项（其中包括软件和目标选择）的值。可允许脚本检索除了由其他环境变量明确提供的命令选项和值之外的所有命令选项和值。例如，当 **SW_SESSIONS_OPTIONS** 指向的文件可用于 *request* 脚本时， *targets* 选项将包含为此命令指定的所有目标的 *software_collection_specs* 列表。当 **SW_SESSIONS_OPTIONS** 指向的文件可用于其他脚本时， *targets* 选项将包含运行脚本的目标的单个 *software_collection_spec* 。

SW_SOFTWARE_SPEC

此变量包含当前产品或文件集的完全限定的软件规范。此软件规范允许唯一标识产品或文件集。

影响由 **swinstall** 和 **swremove** 运行脚本的其他环境变量:

SW_CONFIG_AFTER_REBOOT

此变量只能由 **configure** 脚本读取。如果设置为任何值，则表示在系统启动期间 **configure** 脚本由 **swconfig** 命令调用。此变量由 **/sbin/init.d/swconfig** 系统启动脚本设置。

SW_DEFERRED_KERNBLD

仅适用于 **swinstall** 。通常不设置此变量。如果设置了此变量，则准备系统文件 **/stand/system** 的必要操作无法在 *postinstall* 脚本内部完成，而是必须由 *configure* 脚本完成。将软件安装在 **/** 以外的位置时会发生此类情况，例如对于群集客户端系统。此变量只能由内核文件集的 *configure* 和 *postinstall* 脚本读取。 **swinstall** 命令将这些环境变量设置为用于内核准备和生成脚本。

SW_INITIAL_INSTALL

仅适用于 **swinstall** 。通常不设置此变量。如果设置此变量，则运行 **swinstall** 会话作为初始系统软件安装的后端（“冷”安装）。

SW_KERNEL_PATH

仅适用于 **swinstall** 。内核的路径。缺省值为 **/stand/vmunix** ，由 **swagent** 选项或 **kernel_path** 定义。

SW_SESSION_IS_KERNEL

指示是否为当前安装（或删除）会话调度内核构建。 **TRUE** 值表示为内核构建调度选定的内核文件集，且需要更改 **/stand/system** 。空值表示不调度内核构建，且不需要更改 **/stand/system** 。

此变量的值始终等于 **SW_SESSION_IS_REBOOT** 的值。

SW_SESSION_IS_REBOOT

指示是否为选定删除的文件集调度重新引导。由于所有 HP-UX 内核文件集同时也是重新引导文件集，因此此变量的值始终等于 **SW_SESSION_IS_KERNEL** 的值。

SW_SESSION_IS_UPDATE

值 **1** 表示在操作系统更新期间，SD 命令由 **update-ux** 命令调用。此变量由 **update-ux** 命令设置。

SW_SYSTEM_FILE_PATH

仅适用于 **swinstall** 。定义内核系统文件的路径。缺省值为 **/stand/system** 。

信号

SD 命令捕获信号 SIGQUIT、SIGINT 和 SIGUSR1。如果收到这些信号，此命令将输出消息，向代理发送一个远程过程调用 (RPC) 以结束，然后退出。

代理将忽略 SIGHUP、SIGINT 和 SIGQUIT。它在收到 SIGTERM、SIGUSR1 或 SIGUSR2 后立即正常退出。终止代理可能使系统软件损坏，因此仅在绝对必要的时候才这样做。请注意，终止 SD 命令后，代理在完成正在运行的任务后才终止。

守护程序将忽略 SIGHUP、SIGINT 和 SIGQUIT。它在收到 SIGTERM 和 SIGUSR2 后立即正常退出。收到 SIGUSR1 后，它在退出前会等待结束软件仓库会话中的复制或删除，因此在必要时可以注册或取消注册软件仓库。在等待过程中请求启动新会话将被拒绝。

以下段落仅适用于 *swconfig*、*swcopy*、 *swinstall*、*swremove* 和 *swverify*。

对于 SIGUSR1，此命令向代理发送 RPC 以立即退出，如同代理收到 SIGTERM 信号一样。SIGUSR1 发送到 SD 控制器时，该控制器将关闭具有 SD B.11.01 版或更高版本的目标代理，然后关闭命令本身。目标代理收到关闭 RPC 的信号时，它将对信号 15 (SIGTERM) 调用自己的处理程序，类似于目标计算机上的超级用户在目标代理进程上使用 **kill** 命令时发生的情况。

锁定

SD 命令使用通用锁定机制读取和修改根目录和软件仓库。此机制允许根目录或软件仓库上有多个读取程序，但是仅允许有一个写入程序。

可以如下方式限制在修改（备用）根目录中的软件的同时修改 SD 命令：在

var/adm/sw/products/swlock

文件（相对于根目录，，如 **/var/adm/sw/products/swlock** ）中使用 *fcntl*(2) 锁定。

可以如下方式限制在修改软件仓库中的软件的同时，修改 SD 命令：在

catalog/swlock

（相对于软件仓库目录，例如 **/var/spool/sw/catalog/swlock** ）文件中使用 *fcntl*(2) 锁定。

所有命令使用上面提到的 **swlock** 文件对根目录和软件仓库设置 *fcntl*(2) 读取锁定。设置读取锁定后，它将防止其他 SD 命令执行修改（即防止设置写入锁定）。

如果 SD 进程永久性终止，且没有运行其他 SD 代理，则可以删除 **swlock** 文件来解除锁定根目录或软件仓库。

返回值

每个 SD 命令调用将返回：

0 sw<task> 成功完成。

1 针对所有 *target_selections* 的 sw<task> 均失败。

2 针对某些 *target_selections* 的 sw<task> 失败。

诊断信息

swconfig 、 **swcopy** 、 **swinstall** 、 **swmodify** 、 **swpackage** 、 **swremove** 以及 **swverify** 命令均支持预览模式，在预览模式下，将通过分析每个 *target_selection* 来处理操作，然后在执行实际任务前退出。

可以使用 **sd** 交互式界面或 **swjob** 命令来查看任意作业或控制器的当前状态以及目标日志文件。

标准输出

如果是非交互式，则命令将为重要事件写入消息。这些事件包括：

- 开始和结束任务消息，
- 在每个主机上启动任务的消息，以及
- 在每个主机上完成任务的消息。

设置 **verbose** 选项后，还将向标准输出发送有关任务的摘要消息。

标准错误

如果是非交互式，命令还将为以下重要错误事件写入消息：

- 每个主机分析失败的消息以及
- 每个主机实际任务失败的消息。

日志记录

所有命令将记录调用命令的主机上的主要事件。它们在与每个 *target_selection* 关联的 **swagent** 日志中记录详细事件。

Command Log

命令将消息记录到 **/var/adm/sw/sw<task>**.log 文件（可以通过修改 **logfile** 选项指定其他日志文件）。

Target Log

swagent 进程在每个 *target_selection* 上执行实际的 **swacl** 、 **swconfig** 、 **swcopy** 、 **swinstall** 、 **swremove** 和 **swverify** 操作。对于针对目标根对象执行的操作， **swagent** 会将消息记录到根目录（如 **/** 或备用根目

录）下的文件 **var/adm/sw/swagent.log** 。对于针对目标软件仓库对象执行的操作， **swagent** 会将消息记录到软件仓库目录（例如 **/var/spool/sw** ）下的文件 **swagent.log** 。

在主机上运行的 **swagentd** 将事件记录至文件 **/var/adm/sw/swagentd.log** 。

Source Depot Audit Log

如果源计算机和目标计算机都已升级到 SD B11.00.00 版或更高版本，则源软件仓库计算机上的系统管理员可以跟踪 哪些用户从源计算机上的软件仓库上取出 哪个软件，以及 何时取出软件。有关详细信息，请参考 *swagent*(1M) 中的 **source_depot_audit** 选项。

文件

/dev/rmt/0m

缺省源磁带位置（请注意，SD 可以读取 **tar** 和 **cpio** 磁带软件仓库）。

/etc/fstab

应挂接的卷列表。

$HOME/.swdefaults

包括某些或所有 SD 选项的用户特定的缺省值。如果此文件不存在，则 SD 将在 **$HOME/.sw/defaults** 中查找用户特定的缺省值。

$HOME/.sw/defaults.hosts

包含用户特定的缺省管理主机列表。

$HOME/.sw/defaults.patchfilters

包含用户特定的缺省修补软件过滤器列表。

$HOME/.sw/sessions/

包含 SD 命令自动保存的会话文件，或用户明确保存的会话文件。

$HOME/.sw/software/

包含用户明确保存的软件文件。

$HOME/.sw/targets/

包含用户明确保存的目标文件。

/usr/lbin/swagent

SD 代理。

/usr/lib/nls/$LANG/sw*.cat

SD 消息清单。

/usr/lib/sw/help/

包含 SD GUI 联机帮助工具使用的帮助文件的目录。

/usr/lib/sw/sys.defaults
包含当前 SD 选项（及其缺省值）的主列表。

/usr/lib/sw/ui/
包含 SD 图形用户界面 (GUI) 使用的说明文件的目录。

/usr/newconfig/var/adm/sw/
包含 SD 产品附带的可配置数据的目录，这些数据基于现有配置有条件地复制到 **/var/adm/sw/** 。

/usr/sbin/sw*
SD 命令。

/var/adm/sw/
包含 SD 所有可配置（和不可配置）数据的目录。此目录也是日志文件所在的缺省位置。

/var/adm/sw/defaults
包含某些或所有 SD 选项的当前系统级缺省值。

/var/adm/sw/defaults.hosts
包含系统级缺省管理主机列表。

/var/adm/sw/defaults.patchfilters
包含系统级缺省修补软件过滤器列表。

/var/adm/sw/getdate.templ
包含调度作业时使用的日期/时间模板集。

/var/adm/sw/host_object
存储本地主机上已注册软件仓库列表的文件。

/var/adm/sw/products/
已安装产品数据库 (IPD)，系统上安装的所有产品的清单。

/var/adm/sw/queue/
包含有关由 SD 命令启动的所有活动和已完成作业（如安装、删除以及其他作业）的信息的目录。

/var/adm/sw/security/
包含系统本身的 ACL、模板 ACL 和用于验证远程请求的密钥文件的目录。

/var/adm/sw/target_hosts
由 **swinstall** 或 **swcopy** 进程创建的缓存文件，其中包含目标主机名及其相关的 *uname* 属性。

/var/spool/sw/
源和目标软件仓库的缺省位置。

/usr/lib/sw/examples/
包含示例软件仓库和示例 swpackage 数据的目录。

作者

Software Distributor 由 HP 开发。 **swagent** 、 **swcopy** 、 **swinstall** 、 **swlist** 和 **swpackage** 由 HP 和 Mark H. Colburn 联合开发（请参阅 *pax*(1) ）。

另请参阅

install-sd(1M) 、 swacl(1M) 、 swagentd(1M) 、 swask(1M) 、 swconfig(1M) 、 swcopy(1M) 、 swinstall(1M) 、 swjob(1M) 、 swlist(1M) 、 swmodify(1M) 、 swpackage(1M) 、 swreg(1M) 、 swremove(1M) 、 swverify(1M) 、 swpackage(4)、 sd(4)。

《Software Distributor 管理指南》（位于 **http://docs.hp.com/zh_cn/** 网站）。

SD 客户网站，网址为 **http://docs.hp.com/en/SD/**。

名称

secure_sid_scripts - 控制是否接受脚本上的 setuid 和 setgid 位

值

保证安全

0

缺省值

1

允许值

0-1

建议值

0-1

说明

此可调参数控制可执行脚本上的 **setuid** 和 **setgid** 位是否具有任何影响。接受脚本上的 **set*id** 会使系统易于受到恶意用户的攻击。

此变量的缺省值为 1，说明 **set*id** 位将被 *execve*(2) 系统调用忽略，以获得更高的安全性。为了获得与旧的版本的兼容性，可以在损害安全性的情况下将此值设置为 0。 HP 强烈建议不要更改此可调参数的值，除非有紧急需要必须这样做。

在执行带有 **set*id** 位的脚本时，内核生成下列错误消息到终端控制和系统日志。（要查看错误消息，请使用 *dmesg*(1M) 或检查 **/var/adm/syslog/syslog.log** ）。

Warning: Ignoring set*id bit on *program_name* **as the tunable secure_sid_scripts is set.**

更改此可调参数的人员

管理员。

更改限制

对此可调参数的更改将在更改后启动新脚本时生效。

何时应更改此可调参数的值?

此可调参数控制操作模式而不是数据结构大小和限制时。对系统的合适设置取决于您更看重安全性还是兼容性。

0 值与先前的 HP-UX 发行版兼容，但也缺乏安全性。

1 值提供安全性，防止使用 **set*id** 脚本的竞争状态 (race condition) 攻击。

更改此值的负面影响

此可调参数仅控制设置了 **set*id** 位的可执行脚本（非程序）。 HP-UX 不提供任何这样的脚本。如果客户希望使用 **set*id** 脚本，可以使用第三方应用程序，例如 **suidperl** 或 **sudo** 。另外，可以在运行具有相应权限的 Shell 脚本的简单 C 程序中包装 Shell 脚本：

```
#include <unistd.h>
```

```
#include <stdlib.h>
#include <string.h>
#define SETUID_SCRIPT "/usr/local/bin/cdeject"

int main(int argc, char *const argv[])
{
  if (strcmp(argv[1], SETUID_SCRIPT) == 0) {
    execv(argv[1], argv+1);
    perror(argv[0]);
  } else {
    fprintf(stderr, "%s is not a known setuid script\n",
        argv[1] ? argv[1] : "unspecified-script" );
  }
  exit(1);
}
```

应该同时更改的其他可调参数值

无。

警告

无。所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

文件

/var/adm/syslog/syslog.log

作者

secure_sid_scripts 由 HP 开发。

另请参阅

chmod(1)、 execve(2)、 kctune(1M)。

名称

sema - 引导时启用或禁用 System V IPC 信号量

说明

sema 可调参数已过时。始终启用 System V IPC 信号量子系统。

概述

System V IPC 包含一些机制，通过这些机制任意进程可以发送和接收数据信息、共享虚拟地址空间并使用信号量进行同步执行。

System V 信号量是用户进程同步或串行化访问通用数据和资源的同步方法。每个信号量“集”都有一个 ID，但每个集都可以包含一个或多个独立的信号量。

信号量集由 **semget()** 使用应用程序指定的“关键字”创建或初始化访问，该关键字可通过 **ftok()** 生成。集上的操作通过 **semop()** 和 **semctl()** 来执行。信号量集通过 **semctl()** 或 **ipcrm** 命令删除。

警告

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

sema 由 AT&T 开发。

另请参阅

ipcs(1)、 ipcrm(1)、 semget(2)、 semop(2)、 semctl(2)、 semaem(5)、 semmni(5)、 semmns(5)、 semmnu(5)、 semmsl(5)、 semume(5)、 semvmx(5)、 sysv_hash_locks(5)。

名称

semaem - adjust-on-exit 最大值

值

无故障

16384

缺省值

16384

允许值

最小值： **0**

最大值： **semvmx** 或小于或等于 **32767** 。

说明

semaem 可调参数为任意一个信号量指定了最大累加的“撤消”值，如同更改了任何一个单独的进程。也就是说，对于指定了 **SEM_UNDO** 的信号量，允许一个进程达到 **semaem** 个未完成的增加或减少操作。

如果应用程序尝试超过此限制，它将接收到一个来自 **semop()** 的 [ERANGE] 错误。

有关 System V 信号量的详细信息，请参考 *sema*(5) 联机帮助页的概述一节。

应该由谁来更改此可调参数?

任何人。

对于更改的限制

对此可调参数的更改将在下次重新引导时生效。

何时应增加此可调参数的值?

当希望应用程序拥有大量的信号量增加或减少操作、或大串没有减少操作介于其间的连续增加操作时、或大串没有增加操作介于其间的连续减少操作时，需要“撤消”语义。

何时应降低此可调参数的值?

当应用程序不再需要增加的值时。

同时应更改哪些其他可调参数的值?

所有的 System V 信号量可调参数都是相互关联的，而且不应该作为独立变量处理。这些可调参数必须作为一个系统来评估，以确保它们能反映应用程序的要求。信号量可调参数包括 **semaem** 、 **semmni** 、 **semmns** 、 **semmnu** 、 **semmsl** 、 **semume** 、 **semvmx** 和 **sysv_hash_locks** 。具体而言， **semvmx** 和 **semaem** 值可能需要一起调整。

警告

所有 HP-UX 内核可调参数都是特定于各个发行版的。未来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺

省值或建议的值。有关安装对可调参数值的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

semaem 由 AT&T 开发。

另请参阅

semop(2)、 sema(5)、 semmni(5)、 semmns(5)、 semmnu(5)、 semmsl(5)、 semume(5)、 semvmx(5)、sysv_hash_locks(5)。

名称

semmni - System V IPC 系统级信号量标识符的数量

值

无故障

2048

缺省值

2048

允许值

最小值： **2**

最大值： **semmns** 。

说明

semmni 可调参数指定了可在任何给定的时间存在的 System V IPC 系统级信号量集（和标识符）的最大数量。每个 **semget()** 系统调用将返回一个标识符 (ID)，以创建一个包含一个或多个（最多 **semmsl** ）信号量的新集。

当先前已经分配所有的 ID 时，如果应用程序尝试创建信号量集，那么它将接收到一个来自 **semget()** 的 [ENOSPC] 错误。

有关 System V 信号量的详细信息，请参考 *sema*(5) 联机帮助页的概述一节。

应该由谁来更改此可调参数？

任何人。

对于更改的限制

对此可调参数的更改将在下次重新引导时生效。

何时应增加此可调参数的值？

如果要求系统中有更多唯一信号量 ID 总数，或者一个或多个应用程序要求更多的信号量 ID 时。没有任何理由指定超过 **semmns** 信号量总数的系统级标识符。

何时应降低此可调参数的值？

如果对信号量集 (ID) 的要求显著减少，或需要硬限制应用程序获取更大数量的信号量 ID 时。

同时应更改哪些其他可调参数的值？

所有的 System V 信号量可调参数都是相互关联的，而且不应该作为独立变量处理。这些可调参数必须作为一个系统来评估，以确保它们能反映应用程序的要求。信号量可调参数包括 **semaem** 、 **semmni** 、 **semmns** 、 **semmnu** 、 **semmsl** 、 **semume** 、 **semvmx** 和 **sysv_hash_locks** 。具体而言， **semmni** 可调参数中的更改也可能需要对 **semmns** 可调参数进行更改。 **semmni** 中非常重大的更改也可能需要对 **sysv_hash_locks** 可调参数进行更改。

警告

所有 HP-UX 内核可调参数都是特定于各个发行版的。未来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

semmni 由 AT&T 开发。

另请参阅

semget(2)、 sema(5)、 semaem(5)、 semmns(5)、 semmnu(5)、 semmsl(5)、 semume(5)、 semvmx(5)、 sysv_hash_locks(5)。

名称

semmns - System V IPC 系统级信号量的数量

值

无故障

4096

缺省值

4096

允许值

最小值： **semmni**

最大值： **335534080**

说明

semmns 可调参数指定了可由应用程序分配的单个 System V IPC 系统级信号量的最大总数。信号量被分配在与 ID 关联的“集”中。这样，可以任何方式在 ID 范围内分配信号量，每个 ID 分配一个或多个信号量。

没有任何理由指定小于 **semmni** （标识符的最大数量）的 **semmns** ，因为每个 ID 至少需要一个信号量。

如果应用程序尝试用超过剩余信号量的信号量来创建一个信号量集，它将接收到一个来自 **semget()** 的 [ENOSPC] 错误。

有关 System V 信号量的详细信息，请参考 *sema*(5) 联机帮助页的概述一节。

应该由谁来更改此可调参数?

任何人。

对于更改的限制

对此可调参数的更改将在下次重新引导时生效。

何时应增加此可调参数的值?

如果应用程序要求系统中有更多的信号量总数时。

何时应降低此可调参数的值?

如果对信号量的要求已经减少，或需要硬限制应用程序获取更大数量的信号量时。

同时应更改哪些其他可调参数的值?

所有的 System V 信号量可调参数都是相互关联的，而且不应该作为独立变量处理。这些可调参数必须作为一个系统来评估，以确保它们能反映应用程序的要求。信号量可调参数包括 **semaem** 、 **semmni** 、 **semmns** 、 **semmnu** 、 **semmsl** 、 **semume** 、 **semvmx** 和 **sysv_hash_locks** 。通常， **semmns** 可调参数中的更改不会要求对其他可调参数进行更改。

警告

所有 HP-UX 内核可调参数都是特定于各个发行版的。未来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

semmns 由 AT&T 开发。

另请参阅

semget(2)、 sema(5)、 semaem(5)、 semmni(5)、 semmnu(5)、 semmsl(5)、 semume(5)、 semvmx(5)、 sysv_hash_locks(5)。

名称

semmnu - 对进程的 System V IPC 撤消结构的最大数量

值

无故障

256

缺省值

256

允许值

最小值： **1**

最大值： **(nproc-4)**

说明

semmnu 可调参数指定了可在任何给定的时间拥有未决“撤消”操作的 System V IPC 系统级进程的最大数量。

如果应用程序达到此限制，它将接收到一个来自 **semop()** 的 [ENOSPC] 错误。

有关 System V 信号量的详细信息，请参考 *sema*(5) 联机帮助页的概述一节。

应该由谁来更改此可调参数？

任何人。

对于更改的限制

对此可调参数的更改将在下次重新引导时生效。

何时应增加此可调参数的值？

当使用信号量“撤消”语义的应用程序进程集超过限制时。

何时应降低此可调参数的值？

如果对信号量“撤消”语义的要求显著减少，或需要对更改信号量的失控应用程序进行限制保护时。

同时应更改哪些其他可调参数的值？

所有的 System V 信号量可调参数都是相互关联的，而且不应该作为独立变量处理。这些可调参数必须作为一个系统来评估，以确保它们能反映应用程序的要求。信号量可调参数包括 **semaem** 、 **semmni** 、 **semmns** 、 **semmnu** 、 **semmsl** 、 **semume** 、 **semvmx** 和 **sysv_hash_locks** 。通常， **semmnu** 中的更改与其他可调参数无关。

警告

所有 HP-UX 内核可调参数都是特定于各个发行版的。未来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

引导时内核内存中分配了撤消结构的二维表（每个结构 ~8 字节）。无论是否被使用，都会保留此部分内存。

该表的两个维是 **semume** 和 **semmnu** （每个进程撤消结构的最大数）。因此内存需求大约为这两个可调参数及结构大小的乘积。请小心设定这两个可调参数，由于乘积的效应，它们会对内存利用率产生很大的影响。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

semmnu 由 AT&T 开发。

另请参阅

semop(2)、 sema(5)、 semaem(5)、 semmns(5)、 semmni(5)、 semmsl(5)、 semume(5)、 semvmx(5)、 sysv_hash_locks(5)。

名称

semmsl - 每个标识符 System V IPC 信号量的最大数量

值

无故障

2048

缺省值

2048

允许值

最小值： **1**

最大值： **10240**

说明

semmsl 可调参数指定了每个信号量识别符 (ID) 单个 System V IPC 信号量的最大数量。

如果应用程序尝试超过此限制，它将接收到一个来自 **semget()** 的 [EINVAL] 错误。

有关 System V 信号量的详细信息，请参考 *sema*(5) 联机帮助页的概述一节。

应该由谁来更改此可调参数？

任何人。

何时应增加此可调参数的值？

如果应用程序需要每个 ID 更多的信号量时。

何时应降低此可调参数的值？

如果应用程序对每个 ID 信号量的需求已经降低，或者需要防止遭受不良行为的应用程序时。

同时应更改哪些其他可调参数的值？

所有的 System V 信号量可调参数都是相互关联的，而且 不应该作为独立变量处理。这些可调参数必须作为一个系统来评估，以确保它们能反映应用程序的要求。信号量可调参数包括 **semaem** 、 **semmni** 、 **semmns** 、 **semmnu** 、 **semmsl** 、 **semume** 、 **semvmx** 和 **sysv_hash_locks** 。具体而言， **semmsl** 可调参数中的更改可能需要 **semmns** 对可调参数进行更改。

警告

所有 HP-UX 内核可调参数都是特定于各个发行版的。未来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

在 HP-UX 11i v1 之前，每个 ID 信号量的最大数量为头文件 **sys/sem.h** 中无正式文件的常量 2048。通常应用程序不依赖该限制。对那些以旧的常量编译的应用程序，高于 2048 的 **semmsl** 可调参数值通常不会出现问题。然而， **semmsl** 可调参数值低于 2048 可能导致其他这样的应用程序失败。应该重新编码以从 *pstat_getipc*(2) 中获得当前的 **semmsl** 值。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺

省值或建议的值。有关安装对可调参数值的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

semmsl 由 AT&T 开发。

另请参阅

semget(2)、 sema(5)、 semaem(5)、 semmns(5)、 semmni(5)、 semmnu(5)、 semume(5)、 semvmx(5)、 sysv_hash_locks(5)。

名称

semume - 每个进程 System V IPC 撤消条目的最大数量

值

无故障

100

缺省值

100

允许值

最小值： **1**

最大值： **semmns**

说明

semume 可调参数指定了单个进程可以拥有未完成（非零）“撤消”操作的 System V IPC 信号量的最大值。

如果应用程序达到此限制，它将接收到一个来自 **semop()** 的 [EINVAL] 错误。

有关 System V 信号量的详细信息，请参考 *sema*(5) 联机帮助页的概述一节。

应该由谁来更改此可调参数?

任何人。

对于更改的限制

对此可调参数的更改将在下次重新引导时生效。

何时应增加此可调参数的值?

如果应用程序需要每个进程更多的信号量撤消操作时。

何时应降低此可调参数的值?

如果每个进程对信号量操作的需求显著减少，或需要对更改信号量的失控应用程序进行限制保护时。

同时应更改哪些其他可调参数的值?

所有的 System V 信号量可调参数都是相互关联的，而且不应该作为独立变量处理。这些可调参数必须作为一个系统来评估，以确保它们能反映应用程序的要求。信号量可调参数包括 **semaem** 、 **semmni** 、 **semmns** 、 **semmnu** 、 **semmsl** 、 **semume** 、 **semvmx** 和 **sysv_hash_locks** 。通常， **semume** 中的更改与其他可调参数无关。然而，如果希望单独的应用程序进程使用更少或更多的具有“撤消”语义的信号量，则对 **semmsl** 或 **semmni** 的更改可能需要对 **semume** 进行更改。

警告

所有 HP-UX 内核可调参数都是特定于各个发行版的。未来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

引导时内核内存中分配了撤消结构的二维表（每个结构 ~8 字节）。无论是否被使用，都会保留此部分内存。

该表的两个维是 **semume** 和 **semmnu** （每个进程撤消结构的最大数）。因此内存需求大约为这两个可调参数及结

构大小的乘积。请小心设定这两个可调参数，由于乘积的效应，它们会对内存利用率产生很大的影响。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

semume 由 AT&T 开发。

另请参阅

semop(2)、 sema(5)、 semaem(5)、 semmni(5)、 semmnu(5)、 semmsl(5)、 semmns(5)、 semvmx(5)、 sysv_hash_locks(5)。

名称

semvmx - 任意单个 System V IPC 信号量的最大值

值

无故障

32767

缺省值

32767

允许值

最小值： **1**

最大值： **65535**

说明

semvmx 可调参数指定了任何给定 System V IPC 信号量可以拥有的最大值。

如果应用程序尝试超过此限制，它将会接收到一个来自 **semop()** 或 **semctl()** 的 [ERANGE] 错误。

有关 System V 信号量的详细信息，请参考 *sema*(5) 联机帮助页的概述一节。

应该由谁来更改此可调参数?

任何人。

对于更改的限制

对此可调参数的更改将在下次重新引导时生效。

何时应增加此可调参数的值?

如果应用程序需要更大的信号量值时。

何时应降低此可调参数的值?

如果对单个信号量值的需求减少，或需要对更改信号量的失控应用程序进行限制保护时。

同时应更改哪些其他可调参数的值?

所有的 System V 信号量可调参数都是相互关联的，而且 不应该作为独立变量处理。这些可调参数必须作为一个系统来评估，以确保它们能反映应用程序的要求。信号量可调参数包括 **semaem** 、 **semmni** 、 **semmns** 、 **semmnu** 、 **semmsl** 、 **semume** 、 **semvmx** 和 **sysv_hash_locks** 。当更改 **semvmx** 时请考虑对 **semaem** 进行更改。

警告

所有 HP-UX 内核可调参数都是特定于各个发行版的。未来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

semvmx 由 AT&T 开发。

另请参阅

semop(2)、 semctl(2)、 sema(5)、 semaem(5)、 semmni(5)、 semmns(5)、 semmnu(5)、 semmsl(5)、 semume(5)、 sysv_hash_locks(5)。

已过时

名称

sendfile_max - sendfile 使用的缓冲区缓存页的最大数量 - 已过时的内核可调参数

说明

此可调参数已过时并已被删除。请不要对此可调参数进行任何变更，因为它将不会对内核产生任何影响。在之前的发行版中，此可调参数用于限制 **sendfile()** 系统调用可以使用的 HP-UX 缓冲区缓存页数。 sendfile 操作不再使用 HP-UX 这一传统的缓冲区缓存页，并且此可调参数已过时。请参考文件缓冲区可调参数（请参阅 *fcache_seqlimit_file*(5)）来控制 **sendfile()** 和其他文件系统操作对物理内存的使用。

作者

sendfile_max 由 HP 开发。

另请参阅

fcache_seqlimit_file(5)。

名称

shmem - 启用或禁用 System V 共享内存

说明

shmem 可调参数已过时。始终启用 System V IPC 共享内存子系统。

共享内存是一种有效的进程间通信 (IPC) 机制。一个进程创建共享内存段并将其附加到它的地址空间。然后，任何希望通过共享内存段与此进程进行通信的进程也会将此共享内存段附加到它们相应的地址空间。附加后，进程便可以根据附加时指定的权限对该段进行读取或写入操作。

警告

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

shmem 由 HP 开发。

另请参阅

shmmax(5)、 shmmni(5)、 shmseg(5)。

名称

shmmax - System V 共享内存段的最大大小（以字节为单位）

值

缺省值

1 GB

允许值

最小值： **2048**

最大值： **0x40000000000**

说明

共享内存是一种有效的进程间通信 (IPC) 机制。一个进程创建共享内存段并将其附加到它的地址空间。然后，任何希望通过共享内存段与此进程进行通信的进程也会将此共享内存段附加到它们相应的地址空间。附加后，进程便可以根据附加时指定的权限对该段进行读取或写入操作。

此可调参数为系统内部这样的段设置了最大大小，从 HP-UX lli 开始是动态设置的。

应该由谁来更改此可调参数？

任何人。

对于更改的限制

对此可调参数的更改将立即生效。

何时应增加此可调参数的值？

如果 **shmmax** 在最大值之下，且用户程序正尝试 **shmget** 大于当前值的段并接收到一个 [EINVAL] 错误消息时，应该增加该可调参数的值。

增加此值的副作用是什么？

唯一的作用是该用户程序可以使用 **shmget** 获得更大的段。

何时应降低此可调参数的值？

如果希望强制用户代码上的行为，从而限制 System V 段的最大大小。

降低此值的副作用是什么？

无。

同时应更改哪些其他可调参数的值？

应该考虑 **shmseg** 和 **shmmni** ，因为强制较小的段可能导致用户代码尝试创建更多的段来完成任务。

警告

所有 HP-UX 内核可调参数都是特定于各个发行版的。未来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系

统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

shmmax 由 HP 开发。

另请参阅

shmem(5)、 shmmni(5)、 shmseg(5)。

名称

shmmni - 系统中 System V 共享内存段标识符的数量

值

缺省值

400 个标识符

允许值

最小值： **3**

最大值： **32768**

说明

共享内存是一种有效的进程间通信 (IPC) 机制。一个进程创建共享内存段并将其附加到它的地址空间。然后，任何希望通过共享内存段与此进程进行通信的进程也会将此共享内存段附加到它们相应的地址空间。附加后，进程便可以根据附加时指定的权限对该段进行读取或写入操作。

此可调参数有效地设置了系统级可创建的唯一段的数量，因为每个段都由内核分配一个标识符。标识符只是内核生成的一个引用，这样任何用户进程都可以使用一个简单的整数来请求共享特定的段，且让内核确定该整数所对应的段。

更改此可调参数的人员

任何用户。

更改限制

对此可调参数的更改立即生效。

尝试降低 **shmmni** 到当前使用中标识符的数量以下或到当前 **shmseg** 值以下将导致生成 [EINVAL] 错误消息。

在没有足够的内存可供内核创建所需结构时，尝试增加 **shmmni** 的值将导致生成 [ENOMEM] 错误消息。

应该何时增加此可调参数的值

如果 System V 共享内存的用户在进行 **shmget()** 调用时接收到 [ENOSPC] 错误信息，则应该增加 **shmmni** 。

增加此值的负面影响

内核内存使用量将轻微增加，因为用以跟踪段的数据结构是基于此可调参数分配的。

应该何时降低此可调参数的值

如果内核内存非常紧张，或已经知道将只需要很少的段，则可以通过减少此可调参数节省少量的内核内存，并因此减少与之关联的数据结构的内存使用量。

降低此值的负面影响

将轻微减少内核内存使用量。

应该同时更改的其他可调参数值

应该考虑 **shmmax** 和 **shmseg** 。应该以与 **shmmni** 同样的方式更改 **shmseg** ，因为降低段的总数而增加每个进程可用的数量仅在希望少数进程使用所有段时有意义。

shmmax 更加复杂且任何对其的更改实际上都取决于所需要的效果。请在更改此可调参数前参考 *shmmax*(5) 联机帮助页，以了解详细信息。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

shmmni 由 HP 开发。

另请参阅

shmem(5)、 shmmax(5)、 shmseg(5)。

名称

shmseg - 每个进程 System V 共享内存段的最大数量

值

缺省值

300 个段

允许值

最小值： **1**

最大值：任何小于或等于 **shmmni** 的值。

说明

共享内存是一种有效的进程间通信 (IPC) 机制。一个进程创建共享内存段并将其附加到它的地址空间。然后，任何希望通过共享内存段与此进程进行通信的进程会将此共享内存段附加到它们相应的地址空间。附加后，进程便可以根据附加时指定的权限对该段进行读取或写入操作。

此可调参数设置了每个进程可以附加的段的数量上限。

应该由谁来更改此可调参数？

任何人。

对于更改的限制

对此可调参数的更改将立即生效。

何时应增加此可调参数的值？

如果用户进程达到了它们的限制，需要更多的段，且当前值小于系统中段的总数 (shmmni)。

增加此值的副作用是什么？

单个进程将能够获得更多的段，可能使另一个以前能获得其所需要的所有段的进程缺乏段。在这种情况下，如果 **shmmni** 低于其最大值，则应该增加它。

何时应降低此可调参数的值？

仅在要对用户进程强制段策略或如果一个失控的进程正在扰乱全局段池时，才应降低此可调参数。否则，保持最大值高于通常使用量是无害的。

降低此值的副作用是什么？

无。

同时应更改哪些其他可调参数的值？

如前所述，应该考虑 **shmmni** 。

警告

所有 HP-UX 内核可调参数都是特定于各个发行版的。未来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺

省值或建议的值。有关安装对可调参数值的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

shmseg 由 HP 开发。

另请参阅

shmmni(5)。

名称

signal： signal.h - 信号的说明

概要

#include <signal.h>

说明

头文件 **<signal.h>** 定义了下列符号常量，其中每个常量扩展为一个不同的常量表达式，类型为：

void (*)(int)

其值不与任何可声明函数匹配。

SIG_DFL 缺省信号处理请求。

SIG_ERR 出错情况下由 **signal()** 返回的值。

SIG_HOLD 保留信号请求。

SIG_IGN 忽略信号请求。

下列数据类型通过 **typedef** 定义：

sig_atomic_t 对象的整数类型，即使在异步中断情况下也可作为一个原子实体进行访问

sigset_t 用于表示信号集的对象的整数类型或结构类型。

pid_t 如 **<sys/types.h>** 中所述。

该头文件还声明了一些用于引用出现在系统中的信号的常量。这里定义的信号以字母 **SIG** 开头。每个信号具有不同的正整数值。 0 值将保留用作空信号（请参阅 *kill*(2) ）。系统中可能存在附加的与实现相关的信号。

HP-UX 支持多个信号接口（请参阅 *sigaction*(2) 、 *signal*(2) 和 *sigset*(3C) ），通过这些接口，进程可指定收到信号时采取的操作。

下列信号在所有实现中均受支持。缺省操作 A（异常中止）、C（继续）、I（忽略）、S（停止）和 T（异常终止）在下表后进行了说明。

信号	缺省操作	说明
SIGABRT	A	进程异常中止信号。 与 **SIGIOT** 相同。
SIGALRM	T	报警时钟。
SIGBUS	A	访问某内存对象的未定义的部分。
SIGCHLD	I	子进程终结或停止。 与 **SIGCLD** 相同。
SIGCLD	I	与 **SIGCHLD**（请参阅下面的“警告”一节）相同。

SIGCONT	C	如果停止，则继续执行。
SIGEMT	A	软件生成的信号
SIGFPE	A	错误的算术运算。
SIGHUP	T	挂起。
SIGILL	A	非法指令。
SIGIO	I	异步 I/O 信号；请参阅 *select*(2)。 与 **SIGPOLL** 相同。
SIGIOT	A	进程异常中止信号。与 **SIGABRT** 相同。
SIGINT	T	终端中断信号。
SIGKILL	T	强行终止（无法捕获或忽略）。
SIGLOST	A	文件锁丢失（NFS 文件锁定）。
SIGPIPE	T	在管道上写入而不被读取。
SIGPOLL	T	可轮询的事件。
SIGQUIT	A	终端退出信号。
SIGSEGV	A	无效的内存引用。
SIGSTOP	S	停止执行（无法捕获或忽略）。
SIGTERM	T	终止信号。
SIGTSTP	S	终端停止信号。
SIGTTIN	S	尝试读取的后台进程。
SIGTTOU	S	尝试写入的后台进程。
SIGUSR1	T	用户定义信号 1。
SIGUSR2	T	用户定义信号 2。
SIGPOLL	T	可轮询的事件。
SIGPROF	T	配置定时器过期。
SIGPWR	I	电源故障（请参阅下面的“警告”一节）。
SIGSYS	A	错误的系统调用。
SIGTRAP	A	跟踪/断点陷阱。
SIGURG	I	高带宽数据可用于套接字。
SIGVTALRM	T	虚拟定时器过期。
SIGWINCH	I	窗口大小更改；请参阅 *termio*(7)。
SIGXCPU	A	超出 CPU 时间限制。
SIGXFSZ	A	超出文件大小限制。
SIGRTMIN	T	第一个实时信号。
SIGRTMAX	T	最后一个实时信号。

宏 **SIGRTMIN** 和 **SIGRTMAX** 计算为整数表达式，并且指定了一个至少包括 **{RTSIG_MAX}** 个信号编号的范围，将其保留用于应用程序及支持实时信号扩展（请参阅 *sigaction*(2)）。

缺省操作如下所示：

A　进程异常终止。此外，对于某些具有此缺省操作的信号，如果满足下列条件，将在接收进程的当前工作目录中生成一个核心映像文件（请参阅 *core*(4)）。

- 接收进程的有效用户 ID 和实际用户 ID 相同。
- 接收进程的有效组 ID 和实际组 ID 相同。
- 名为 **core** 的常规文件不存在并且可以创建，或者该文件存在并且可以写入。

如果创建了该文件，则其具有以下属性：

- 文件模式为 0600，通过文件创建模式掩码来修改（请参阅 *umask*(2)）。
- 文件用户 ID 与接收进程的有效用户 ID 相同。
- 文件组 ID 与接收进程的有效组 ID 相同。

C　如果进程停止，则继续执行进程；否则将忽略信号。

I　忽略信号。

S　停止进程。当进程停止时，发送到进程的任何其他信号将挂起，直到进程重新启动为止（**SIGKILL** 除外，它会被立即处理）。但是，当进程重新启动时，将会处理未决的信号。当孤立的进程组（请参阅 *glossary*(9)）中的进程收到 **SIGTSTP**、 **SIGTTIN** 或 **SIGTTOU** 信号时，将不会停止进程，这是因为不允许停止孤立进程组中的进程。但是会将 **SIGHUP** 信号发送给进程，同时忽略 **SIGTSTP**、 **SIGTTIN** 或 **SIGTTOU** 信号。

T　异常终止进程。进程将随 **exit()** 的所有结果而被终止，当指定信号使得表示异常终止的状态对 **wait()** 和 **waitpid()** 可用时例外。请参阅 *exit*(2)、 *wait*(2)、 *waitpid*(2)。

头文件提供了 struct **sigaction** 的声明，其中至少包括下列成员：

void	***(*sa_handler)(int)***	收到信号时如何操作
sigset_t	***sa_mask***	信号处理函数执行期间要阻止的信号集。
int	***sa_flags***	特殊标记
void (*) ***(int, siginfo_t *, void *)***	***sa_sigaction***	指向信号的指针处理程序函数或其中一个宏 *SIG_IGN* 或 *SIG_DFL*

由 *sa_handler* 和 *sa_sigaction* 占用的存储空间可能会重叠，一个可移植程序决不能同时使用这二者。

以下为常量声明：

SA_NOCLDSTOP　当子进程停止时，不生成 **SIGCHLD**。

SIG_BLOCK	得到的集合是当前集合与参数 *set* 所指向的信号集所构成的并集。
SIG_UNBLOCK	得到的集合是当前集合与参数 *set* 所指向的信号集的补集所构成的交集。
SIG_SETMASK	得到的集合是参数 *set* 所指向的信号集。 **SA_ONSTACK** 将导致在备用堆栈上发送信号。
SA_RESETHAND	使得在进入信号处理程序时，信号处理状态设置为 **SIG_DFL** 。
SA_RESTART	使得某些函数能够重新启动。
SA_SIGINFO	使得在收到信号时，将额外信息传递给信号处理程序。
SA_NOCLDWAIT	使得在子进程死亡时，实现不会创建死进程。
SA_NODEFER	使得在进入信号处理程序时，不会自动阻止信号。
SS_ONSTACK	进程正在备用信号堆栈上执行。
SS_DISABLE	禁用备用信号堆栈。
MINSIGSTKSZ	信号处理程序的最小堆栈大小。
SIGSTKSZ	备用信号堆栈的缺省大小，以字节为单位。

结构 **ucontext_t** 通过 **typedef** 进行定义，如 **<ucontext.h>** 中所述。

mcontext_t 结构是通过 **typedef** 定义的，如 **<ucontext.h>** 中所述。

头文件 **<signal.h>** 将 **stack_t** 类型定义为至少包括以下成员的结构：

void	****ss_sp***	堆栈基数或指针
size_t	***ss_size***	堆栈大小
int	***ss_flags***	标记

头文件 **<signal.h>** 定义了 **sigstack** 结构，其中至少包括以下成员：

int	***ss_onstack***	当信号堆栈使用时为非零值
void	****ss_sp***	信号堆栈指针

头文件 **<signal.h>** 定义了 **sigevent** 结构，其中至少包括下列成员：

int	***sigev_notify***	通知类型
int	***sigev_signo***	信号编号
union sigval	***sigev_value***	信号值。

成员 *sigev_notify* 指定发生异步事件时要使用的通知机制。下面定义的值用于 *sigev_notify* 成员：

SIGEV_NONE	当有意义的事件发生时，将不发送异步通知。

SIGEV_SIGNAL 当有意义的事件发生时，将为进程生成 *sigev_signo* 中指定的信号。如果为此信号编号设置了 **SA_SIGINFO** ，则将此信号加入进程的信号队列，并且 *sigev_value* 中指定的值将成为所生成信号的 *sigev_value* 部分。如果未为此信号编号设置 **SA_SIGINFO** ，则信号是否加入队列，以及发送的值（如果有的话）均不确定。

成员 *sigev_signo* 指定了要生成的信号。成员 *sigev_value* 是由应用程序定义的值，该值在发送信号时向信号传递捕获函数，或者在接收信号时返回，作为 *siginfo_t* 结构的 *si_value* 成员。

头文件 **<signal.h>** 将 **sigval** 定义为一个联合结构，其中至少包括下列成员：

int	***sival_int***	整数信号值
void *	***sival_ptr***	指针信号值。

头文件 **<signal.h>** 将 **siginfo_t** 类型定义为至少包括以下成员的结构：

int	***si_signo***	信号编号
int	***si_errno***	如果不为零，则为与该信号相关联的 **errno** ，如 **<errno.h>** 中所定义。
int	***si_code***	信号代码
union sigval	***si_value***	信号值
id_t	***si_pid***	发送进程 ID
uid_t	***si_uid***	发送进程的实际用户 ID
void	****si_addr***	出错指令的地址
int	***si_status***	退出值或信号
long	***si_band***	SIGPOLL 的带事件

成员 *si_code* 包含了标识信号起因的代码。下面定义的值用于 *si_code* ：

SI_USER 信号由 **kill()** 发送。如果信号由 **raise()** （请参阅 **kill**(2) ）或类似函数（作为 **kill()** 的实现扩展而提供），则 *si_code* 也可能被设置为 **SI_USER** 。

SI_QUEUE 信号由 **sigqueue()** 发送。

SI_TIMER 信号是由于 **timer_settime()** 设置的定时器过期而生成。

SI_ASYNCIO 信号是通过完成一个异步 I/O 请求而生成。

SI_MESGQ 信号是由于一条消息到达空消息队列而生成。

如果信号不是由这些函数之一或上面列出的事件（例如 **kill()** 、 **raise()** 、 **sigqueue()** 和其他函数）生成，则 *si_code* 将被设置为与上面定义的任何 *si_code* 值都不相等的由实现定义的值（请参阅下面的 代码列）。

如果 *si_code* 是 **SI_QUEUE** 、 **SI_TIMER** 、 **SI_ASYNCIO** 或 **SI_MESGQ** 中的一个，则 *si_value* 将包含由应用程序指定的信号值。否则， *si_value* 的内容将无法确定。

定义下表的“代码”列中指定的宏用作 *si_code* 的值，这些值表示生成信号的特定于信号的原因。

信号	代码	原因
SIGILL	**ILL_ILLOPC**	非法操作码
	ILL_ILLOPN	非法操作数
	ILL_ILLADR	非法寻址模式
	ILL_ILLTRP	非法陷阱
	ILL_PRVOPC	特权操作码
	ILL_PRVREG	特权寄存器
	ILL_COPROC	协处理器错误
	ILL_BADSTK	内部堆栈错误
SIGFPE	**FPE_INTDIV**	整数除以零
	FPE_INTOVF	整数溢出
	FPE_FLTDIV	浮点数除以零
	FPE_FLTOVF	浮点数溢出
	FPE_FLTUND	浮点下溢
	FPE_FLTRES	浮点不精确结果
	FPE_FLTINV	无效的浮点运算
	FPE_FLTSUB	下标越界
SIGSEGV	**SEGV_MAPERR**	地址未映射到对象
	SEGV_ACCERR	映射对象的无效权限
SIGBUS	**BUS_ADRALN**	无效的地址对齐
	BUS_ADRERR	不存在的物理地址
	BUS_OBJERR	对象的特定硬件错误
SIGTRAP	**TRAP_BRKPT**	进程断点
	TRAP_TRACE	进程跟踪陷阱
SIGCHLD	**CLD_EXITED**	子进程已退出
	CLD_KILLED	子进程已异常终止，并且未创建一个核心文件
	CLD_DUMPED	子进程已终止，并且创建了一个核心文件
	CLD_KILLED	子进程已强行终止
	CLD_DUMPED	子进程已异常终止
	CLD_TRAPPED	被跟踪的子进程已捕获
	CLD_STOPPED	子进程已停止
	CLD_CONTINUED	停止的子进程已继续
SIGPOLL	**POLL_IN**	数据输入可用
	POLL_OUT	输出缓冲区可用
	POLL_MSG	输入消息可用

POLL_ERR	I/O 错误
POLL_PRI	高优先级输入可用
POLL_HUP	设备断开连接

实现可能支持此列表中不包含的其他 *si_code* 值，在除了此表中所述情况之外的其他情况下，可能会生成表中所包含的值，还可能包含阻止一些值生成的扩展或限制。对于此列表中所描述的情况，实现不会生成一个与表中所述不同的值。

此外，将可以使用下列特定与信号的信息：

信号	成员	值
SIGILL	***void * si_addr***	出错指令的地址
SIGFP		
SIGSEGV	***void * si_addr***	出错内存引用的地址
SIGBUS		
SIGCHLD	***pid_t si_pid***	子进程 ID
	int si_status	退出值或信号
	uid_t si_uid	发送信号进程的实际用户 ID
SIGPOLL	***long si_band***	**POLL_IN** 、 **POLL_OUT** 或 **POLL_MSG** 的带事件

对于某些实现， *si_addr* 的值可能是不准确的。

以下内容将声明为函数，也可以定义为宏：

```
void  (*bsd_signal(int sig, void (*func)(int)))(int);
int   kill(pid_t pid, int sig);

int   killpg(pid_t pgrp, int sig);
int   raise(int sig);
int   sigaction(int sig, const struct sigaction
           *act, struct sigaction *oact);
int   sigaddset(sigset_t *set, int signo);

int   sigaltstack(const stack_t *ss, stack_t *oss);
int   sigdelset(sigset_t *set, int signo);
int   sigemptyset(sigset_t *set);
int   sigfillset(sigset_t *set);

int   sighold(int sig);
int   sigignore(int sig);
int   siginterrupt(int sig, int flag);
int   sigismember(const sigset_t *set, int signo);

int   sigmask(int signum);
```

void (*signal(int sig, void (*func)(int)))(int);

int sigpause(int sig);

int sigpending(sigset_t *set);

int sigprocmask(int how, const sigset_t *set, sigset_t *oset);

int sigqueue(pid_t pid, int sig, const union sigval value);

int sigrelse(int sig);

void *sigset(int sig, void (*disp)(int)))(int);

int sigstack(struct sigstack *ss,
struct sigstack *oss);

int sigsuspend(const sigset_t *sigmask);

int sigtimedwait(const sigset_t *set, siginfo_t * info,
const struct timespec *timeout);

int sigwait(const sigset_t *set, int *sig);

int sigwaitinfo(const sigset_t *set, siginfo_t * info);

实际应用信息

在接收信号时，如果使用 **signal()** 将操作设置为信号处理程序的地址，则将信号捕获操作重置为 **SIG_DFL** （ **SKGILL** 、 **SIGTRAP** 和 **SIGPWR** 除外）。然后，将执行信号捕获函数。除 **signal()** 之外的信号接口例行程序通常不会重置信号捕获操作。但是， **sigaction()** 提供了指定此行为的办法。

传递给使用 **signal()** 的信号捕获函数的参数为:

int	***sig***	信号编号。
int	***code***	通常由硬件 提供的信息字。
struct sigcontext	****scp***	指向在 **<signal.h>** 中 定义的计算机相关的结构 *sigcontext* 的指针。

传递给使用 **sigaction()** 并且未指定 **SA_SIGINFO** 的信号捕获函数的参数是:

int	***sig***	信号编号。

传递给使用 **sigaction()** 并且指定了 **SA_SIGINFO** 的信号捕获函数的参数是:

int	***sig***	信号编号。
siginfo_t	****siginfop***	有关生成信号的原因的 信息。
void	****contextp***	可以转换为指向 *ucontext_t* 的指针， 以便在中断时引用 线程的环境。

code 可以为零和（或） *scp* 可以为 **NULL** ，这取决于 *sig* 的值。 *code* 与 *scp* 的含义，以及确定它们何时不为零或 **NULL** 的条件与实现相关。 *code* 可能始终为零，而 *scp* 始终为 **NULL** 。

指针 *scp* 、 *siginfop* 和 *contextp* 仅在信号捕获函数的环境中才有效。

可选参数可以从信号捕获函数的参数列表中省略，在这种情况下，信号捕获函数完全与 UNIX 系统 V 相兼容。真正的可移植软件不应使用信号捕获例行程序中的可选参数。

接收进程从信号捕获函数中返回时，将在其中断的位置恢复执行。

在慢速设备（例如终端，但不是本地文件系统上的文件）上执行系统调用（例如 **read()** 、 **write()** 、 **open()** 或 **ioctl()** ）期间，以及由于前面已存在停止进程或死进程，而使得 **pause()** 或 **wait()** 系统调用不会立即返回的过程中，如果捕获到信号，则将执行信号捕获函数，并且中断的系统调用将 –1 返回给调用的进程，同时 **errno** 设置为 [EINTR] 。

如果生成了进程的任一停止信号（ **SIGSTOP** 、 **SIGTSTP** 、 **SIGTTIN** 、 **SIGTTOU** ），则将忽略该进程未决的 **SIGCONT** 信号。相反，如果生成进程的 **SIGCONT** 信号，则将忽略该进程的所有未决的停止信号。如果生成停止进程的 **SIGCONT** 信号，则即使阻止或忽略 **SIGCONT** 信号，该进程也将继续执行。如果阻止但未忽略 **SIGCONT** 信号，则在取消阻止该信号或生成停止信号之前，该信号将保持未决状态。

注释：如果将任一停止信号（ **SIGSTOP** 、 **SIGTSTP** 、 **SIGTIN** 、 **SIGTTOU** ）传递给进程争用范围创建的线程，则通过使用 **pthread_kill()** ，可能不会忽略未决的 **SIGCONT** 信号。同样，如果将 **SIGCONT** 信号传递给进程争用范围创建的线程，通过使用 **pthread_kill()** ，不会忽略未决的停止信号。但是，当将停止和继续信号传递到系统争用范围创建的线程时，通过使用 **pthread_kill()** ，信号处理将继续遵守前一段落中所述的语义。

如果 **exec()** 系统调用失败，并且原始程序已被删除，则将由系统发送 **SIGKILL** 信号。

线程注意事项

线程的信号模型总结如下：

指定阻止发送的信号的信号掩码与每个线程关联。

信号的处理（捕获/忽略/缺省）是进程属性，并且由进程中的所有线程共享。

如果某个信号的信号操作指定为终止、停止或继续，则将分别终止、停止或继续进程中的所有线程。无论信号是指向进程还是指向进程中的特定线程，这种情况都会存在。

由某些与特定线程关联的操作生成的信号（例如无效指针的取消引用），将传递到导致信号生成的线程。这些信号称为同步生成的信号。

由 **kill()** 或某些异步事件（例如终端活动）发送到进程的信号，将被传递到该进程内一个不阻止信号传递的线程；如果有多个符合条件的线程，则应用程序可能无法确定要将信号发送到的线程。如果进程内的所有线程都阻止信号，则信号在进程上保持未决状态，直到线程解除阻止该信号，对信号发出 **sigwait()** 调用或者设置信号配置以忽略该信号。这些信号称为异步生成的信号。

通过使用 **pthread_kill()** ，线程可以将信号传递给同一进程中的特定线程。如果向其传递信号的线程阻止发送信号，则信号在线程中将保持未决状态。

函数 **sigpending()** 将返回进程中和调用线程中未决的信号集的并集。

每个 *PTHREAD_SCOPE_SYSTEM* 线程可以定义一个备用信号处理堆栈。在 *PTHREAD_SCOPE_PROCESS* 中使用备用信号堆栈将会导致不确定的行为。

线程安全注意事项

有关 libc 和 libpthread 接口（非线程安全、取消点、取消安全、异步信号安全和异步取消安全）列表的详细信息，请参考 *thread_safety*(5) 。

警告

信号 **SIGCLD** 和 **SIGPWR** 的行为与以上“说明”中的信号列表中所述信号不同。

关于这些信号的操作修改如下所示:

SIGCLD

在父进程中将关于 **SIGCLD** 的操作设置为 **SIG_IGN** ，可防止调用进程的退出子进程创建一个死进程。如果父进程执行 **wait()** 函数，则调用进程将阻塞，直到调用进程中的所有子进程终止为止。然后， **wait()** 函数返回值 −1 ，同时将 **errno** 设置为 [ECHILD] （请参阅 *wait*(2) ） 。

在当前具有已终止（或死亡）子进程的进程中，如果使用其中一个信号接口例行程序将 **SIGCLD** 的操作设置为捕获（也就是说，提供了函数地址） ，则 **SIGCLD** 信号将立即发送到父进程。

signal() 设置的信号捕获函数在捕获信号时将重置。这些函数通常会重新安装其自身。如果与 **SIGCLD** 关联的函数在调用一个子等待函数（例如 **wait()** 、 **wait3()** 和 **waitpid()**）之前重新安装其自身，将传递另一个 **SIGCLD** 信号，并且立即再次调用处理程序（除非该信号被屏蔽） 。因此，用于 **SIGCLD** 的信号捕获函数仅在它调用了一个子等待函数后才安装其自身（这同样适用于在指定了 *SA_RESETHAND* 或 *SV_RESETHAND* 的情况下，使用 **sigaction()** 设置的信号捕获函数） 。

SIGPWR

电源中断后，当电源恢复并且系统完成了所有必需的重新初始化时，信号 **SIGPWR** 将发送到所有进程。通过捕获（或忽略） **SIGPWR** 信号，进程将重新启动。

希望从电源故障中恢复的应用程序应当捕获 **SIGPWR** ，并且采取任何必要的步骤对自身进行重新初始化。

某些实现不生成 **SIGPWR** 。仅具有永久性存储器的系统可以从电源故障中恢复。

当出现陷阱时，硬件程序计数器计数器不会前进。如果由硬件陷阱生成的信号由用户程序中的信号捕获函数接收到，则从信号捕获函数返回时将重新执行引发陷阱的指令，从而导致重新出现陷阱，除非信号捕获函数对程序流进行更改。例如，可以调用 **longjmp()** 例行程序（请参阅 *setjmp*(3C) ） 。使用 **longjmp()** 可确保软件在不同硬件体系结构上的可移植性。

如果信号捕获函数不接收硬件陷阱生成的信号，即，如果信号被屏蔽或忽略，则可能有两种结果:

1. 重新执行导致陷阱的指令，导致程序无穷地循环。

2. 系统检测到此状态，导致程序终止。

作者

signal 由 HP、 AT&T 和加州大学伯克利分校联合开发。

另请参阅

kill(1)、 init(1M)、 alarm(2)、 exit(2)、 ioctl(2)、 kill(2)、 lseek(2)、 pause(2)、 select(2)、 sigaction(2)、 sigaltstack(2)、 siginterrupt(2)、 signal(2)、 sigpending(2)、 sigprocmask(2)、 sigsuspend(2)、 sigwait(2)、 umask(2)、 wait(2)、 waitid(2)、 abort(3C)、 setjmp(3C)、 sigset(3C)、 core(4)、 thread_safety(5)、 termio(7)、 glossary(9)。

符合的标准

<signal.h>： AES、 SVID2、 SVID3、 XPG2、 XPG3、 XPG4、 FIPS 151-2、 POSIX.1、 ANSI C

名称

sis - 具有 Kerberos 验证和授权的安全 Internet 服务

说明

当与 HP DCE 安全服务、 HP Praesidium/Security 服务器或其他可提供 Kerberos V5 网络验证服务环境的软件产品一起使用时，“安全 **Internet** 服务” (SIS) 将提供网络验证。网络验证确保本地和远程的主机将以一种安全和可信的方式彼此相互标识，并且授权用户在远程主机上使用此服务。

通过传统 Internet 服务（例如 **telnet** 、 **rlogin** 或 **ftp** ）用户可以键入口令，随后将其通过网络传送给远程系统，对远程系统进行访问。口令将不经加密通过网络传送，从而使得观察程序可捕获到包含口令的明文数据包。这已经成为传统 Internet 服务的一个主要安全漏洞。

可选的安全 Internet 服务可取代对应的传统服务，并防止通过网络明文传送用户口令。但是，所有这些服务都不会对会话进行加密，但用于验证服务或授权用户所必需的会话除外。

本联机帮助页假定读者熟悉 Kerberos 术语，该术语通常在用户的 Kerberos V5 网络验证服务环境中提供。此处的目的是描述特定由 SIS 使用的 Kerberos 环境的相关情况。

验证

为使 Kerberos 验证成功，用户必须在 Kerberos 领域内成功登录到系统，并且获得一组凭证。凭证包括凭证授予凭证 (TGT) 和会话密钥。 SIS 客户端将使用 TGT 获取服务凭证，以对网络上的 SIS 守护程序进行访问。如果凭证丢失或者 TGT 无效，则验证将失败，并且将拒绝与 SIS 守护程序的连接。

对于配置到 DCE 单元的系统，可通过 **dce_login** 命令获取凭证。对于配置到 Praesidium/Security 服务器单元的系统，可通过 **dess_login** 命令获取凭证。在基于 Kerberos 的非 DCE 安全环境中，可通过 **kinit** 命令获取凭证。

授权

对于这些服务的每个用户，必须将用户主体配置到密钥分配中心的数据库。通过用户主体，用户可获得服务凭证，该凭证作为 Kerberos 验证机制的一部分，发送到远程服务。如果验证成功，则将用户主体用作 Kerberos 授权机制的一部分。

为使授权成功，必须同时满足下列两个要求：

1. 在远程系统的口令文件中，登录名称必须存在，也就是说，远程帐户必须存在。注释：登录名称是指用户响应登录提示所指定的名称，可能与当前用户名称不同。

2. 下列情况之一必须为真：

   A. 远程帐户的主目录有一个包含用户主体的 **.k5login** 文件。 **.k5login** 文件必须由该帐户拥有，并且仅该帐户具有写入权限（即，权限显示为 **-rw-r--r--** ）。

      注释：在远程系统中，如果 **/etc/krb5/** 目录存在， Kerberos 将忽略远程帐户主目录中的 **.k5login** 文件。

   B. 远程系统在包含用户主体的 **/etc/krb5/** 目录中拥有一个 **.k5login.***login_name* 文件或符号链接。如果 **/etc/krb5/** 目录不存在，Kerberos 将检查远程帐户主目录中的 **.k5login** 文件。如果 **/etc/krb5/** 目录存在，Kerberos 将忽略远程帐户主目录中的 **.k5login** 文件。

.k5login.*login_name* 文件中条目的格式与 **.k5login** 文件中条目的格式相似。 **.k5login.***login_name* 文件（或符号链接）和 **/etc/krb5/** 目录必须由超级用户拥有，并且只有超级用户必须拥有写入权限（即 **-rw-r--r--**）。要授予用户处理 **.k5login.***login_name* 文件的权限，管理员可以创建指向远程帐户主目录中 **.k5login** 文件的 **.k5login.***login_name* 符号链接。

C. 远程系统有一个包含用户主体的授权名称数据库文件 **aname** 。 **aname** 文件应包含用户主体到远程系统上帐户的映射。

D. 用户主体中的用户名与所访问的帐户的用户名相同，并且本地系统和远程系统在同一领域中。

如果授权成功，则用户将不会看见口令提示（当需要使用口令时），并且将成功连接到远程系统。如果验证或授权失败，则将通知用户出现错误，并且不允许其继续进行。

绕过或强制验证/授权

如果验证或授权失败，则可以用一个特殊的命令行选项 (**-P**) 来重新运行服务，以请求非 Kerberos 验证。但是，当需要口令时，口令将以可读取的形式通过网络发送。通常，该特殊命令行选项应当仅用于访问非安全远程系统。

ftp 和 **telnet** 守护程序有一个特殊的命令行选项 (**-A**)，可用于确保拒绝对非安全系统进行访问。

为防止通过 rcp、remsh 或 rlogin 命令进行非安全访问，应对远程系统上的 **inetd.conf** 文件进行编辑，以注释取消用于 **shell** 和 **login** 的条目。

服务

ftp, **ftpd**	文件传输程序
rlogin, **rlogind**	远程登录
telnet, **telnetd**	Telnet 协议的用户接口和服务器
rcp, **remshd**	远程文件复制
remsh, **remshd**	从远程 Shell 执行

疑难解答

为了正确执行 SIS，正确安装、配置和运行安全环境是十分重要的。以下是对此进行验证的快速检查列表：

1. DCE、Praesidium/Security 服务器或 Kerberos 安全系统应在 Kerberos 服务器上运行。 **/etc/services** 文件应包含 Kerberos 端口的条目。
2. 用户的用户主体必须输入到密钥分配中心的数据库。使用适当的工具（例如， **kadmin** 或 HP DCE 的 **dcecp** ）列出数据库，并验证用户是否已配置了用户主体。
3. 本地系统和远程系统上的 Kerberos 配置目录中应包含 **krb.conf** 、 **krb.realms** 和服务器密钥表文件。通常，Kerberos 配置目录将为 **/krb5** ，服务器密钥表文件将命名为 **v5srvtab** 。
4. 必须在本地系统和远程系统上的 **/etc/krb5/.k5login.***login_name* 或 **~/.k5login** 中指定用户主体。 **~/.k5login** 列出了对用户帐户具有访问权限的主体和领域的名称。

或者，安全系统可以在本地系统和远程系统上使用授权名称数据库文件 **aname** 。此文件中的条目将用户主体中的用户名授权给指定的登录名。

验证 **/etc/krb5/.k5login.***login_name* 或 **~/.k5login** 是否存在，是否具有正确的权限（即 **-rw-r--r--** ），并且包括用户主体。或者，使用适当的工具（例如，非 HP DCE 系统上的 **krb5_anadd** ）验证用户主体是否包含在 **aname** 文件中。

注释：在远程系统中，如果 **/etc/krb5/** 目录存在， Kerberos 将忽略远程帐户的主目录中的 **.k5login** 文件。

5. 远程系统上的服务器密钥表文件应包含主机主体。超级用户可通过下列命令验证 v5srvtab 的内容： **klist -k** 。如果 **klist** 支持 **-k** 选项，则请键入此命令来验证是否列出了主机主体。

   或者，如果系统中可使用验证工具 **krbval** ，则请使用命令： **krbval -v** 。

6. 如果在本地系统和远程系统中可使用 **krbval** ，则通过调用它作为本地系统上的客户端应用程序和远程系统上的服务器应用程序，来测试 Kerberos 配置。有关详细信息，请参阅 *krbval*(1M) 。

7. 必须安装 SIS 文件。传统服务已进行保存，用于新服务的文件将链接到原始的、传统的文件名。

诊断信息

除了特定于 Kerberos 的错误消息外，SIS 还有一些多个服务或全部服务常用的与安全性相关的错误消息。这些错误消息可由脚本使用，以检测服务的调用是否已失败。

SIS 客户端报告的错误和警告消息

ERROR! Kerberos authentication failed.

用户尚未获取有效的凭证授予凭证（通过 **kinit** 、 **dce_login** 或 **dess_login** ），或者尚未在该领域的密钥分配中心的数据库中配置有效的主机主体。伴随此错误消息，将有一条指明可能失败原因的更具体的错误消息。

如果用户尝试访问一个非安全远程系统，则也将生成此错误消息。在这种情况下，将在此消息前加上一条消息： **To bypass Kerberos authentication, use the -P option .**

此错误由 ftp、rlogin 和 telnet 报告。

ERROR! Kerberos-specific options are invalid with the -P option.

-P 命令行选项表示不应执行 Kerberos 验证。如果命令行中还指定了其他任何特定于 Kerberos 的选项，则这些选项与此请求相反。

对于 **remsh** 和 **rlogin** ，这意味着 **-P** 选项不能与 **-F** 、 **-f** 或 **-k** 选项一起使用。

对于 **rcp** ，这意味着 **-P** 选项不能与 **-k** 选项一起使用。

对于 **telnet** ，这意味着 **-P** 选项不能与 **-a** 或 **-l** 选项一起使用。

WARNING! Password will be sent in a non-secure manner.

WARNING! Kerberos authentication will be bypassed.

用户已在命令行中指定了 **-P** 选项，以对非安全远程系统进行访问，或者绕过 Kerberos 环境中的错误配置。

在需要口令的情况下， **-P** 命令行选项使得口令以可读形式通过网络发送，这种环境下口令很可能被截取或捕获。

建议用户更正错误配置，并且仅当远程系统是非安全的情况下才可使用 **-P** 选项。

第一个警告由 **ftp** 、 **rlogin** 和 **telnet** 报告。第二个警告由 **rcp** 报告。 **remsh** 可报告任一警告，具体取决于是否需要口令。

在 **syslog** 中由 **SIS** 守护程序报告的错误消息

ERROR! Access denied. Kerberos authentication must succeed.

守护程序以 **-A** 命令行选项开头，以确保拒绝远程系统的非安全访问。除非已针对安全访问配置了本地系统，否则用户无法访问远程系统。

此错误由 **ftpd** 和 **telnetd** 记录。

ERROR! Principal *principal* **(***remote_user*@*remote_host***) logging in as** *local_user* **has no account.**

local_user 没有有效的口令文件条目。

此错误由所有 SIS 守护程序记录。

ERROR! Principal *principal* **(***remote_user*@*remote_host***) logging in as** *local_user* **failed krb5_userok.**

验证成功但授权失败。用户应验证他们的用户名是在 **/etc/krb5/.k5login.***login_name* 或 **~/.k5login** 中列出，或者是在远程系统上的 **aname** 文件中列出。用户的 **~/.k5login** 必须具有正确的权限，并且必须由该用户拥有（即， **-rw-r--r--** ）。

此错误由所有 SIS 守护程序记录。

ERROR! Principal *principal* **(***remote_user*@*remote_host***) logging in as** *local_user* **failed ruserok.**

/etc/hosts.equiv 或 **~/.rhost** 文件丢失，或者未正确设置来授权 *local_user* （请参阅 *ruserok*(3N) ）。

如果 **rlogind** 或 **remshd** 以 **-R** 、 **-r** 或 **-k** 选项开头，则此错误将由它们记录。

另请参阅

ftp(1)、 kdestroy(1)、 kinit(1)、 klist(1)、 krbval(1M)、 rcp(1)、 remsh(1)、 rlogin(1)、 telnet(1)、 dce_intro(1M)、 dce_login(1M)、 dess_login(1M)、 ftpd(1M)、 remshd(1M)、 rlogind(1M)、 telnetd(1M)、 dess(5)。

名称

standards - HP-UX 的 UNIX 标准行为

说明

HP-UX 遵循多种 UNIX 标准。在某些情况下，这些标准会发生冲突。本联机帮助页介绍程序员和用户要使应用程序遵循特定的 UNIX 标准并按照此标准执行而必须使用的方法。

UNIX 标准兼容程序员环境

下表列出了编译应用程序时必须定义的功能测试宏和环境变量。功能测试宏和环境变量 必须在编译应用程序时定义，以使应用程序遵循特定的 UNIX 标准并根据该标准执行。否则，此行为是不明确的。

Standard	Feature Test Macros to be defined during compilation	Environment variable to be set
UNIX 95	_XOPEN_SOURCE_EXTENDED=1	UNIX95 or UNIX_STD=95 or UNIX_STD=1995
UNIX 2003	_XOPEN_SOURCE=600	UNIX_STD=2003

编译程序使用功能测试宏从头文件获得适当的命名空间。编译程序使用环境变量在适当的对象文件中链接到可执行文件。使用环境变量自定义 libc，以使 UNIX 标准匹配各种功能。

如果应用程序已经按缺省 HP-UX 行为或按特定标准进行编译，但是需要更改为特定的 UNIX 标准行为，则请按照上表中指定的标准重新编译应用程序。

要使 HP-UX 命令遵循特定的 UNIX 标准行为，在执行该命令前，应用程序必须按照上表中的指定来设置相应的环境变量。

UNIX 标准兼容用户环境

要启用特定的 UNIX 标准兼容用户环境，请按照上表中的定义设置相应的环境变量。

举例

下面显示了一个应用程序示例。要使系统兼容 UNIX2003 行为，请将 **UNIX_STD** 环境变量设置为 **2003** ，并在编译前定义 **_XOPEN_SOURCE=600**

功能测试宏。

```
$ export UNIX_STD=2003
$ c99 -D_XOPEN_SOURCE=600  foo.c
```

以下示例通过在执行命令前将其中一个环境变量设置为 **UNIX95** 或 **UNIX_STD=95** ，将 **ls** 命令更改为具有 UNIX95 行为。为 UNIX95 设置环境变量有以下三种方法:

```
$ export UNIX95=1
$ ls 选项
```

或者

```
$ export UNIX_STD=95
$ ls 选项
```

或者

```
$ export UNIX_STD=1995
$ ls 选项
```

另请参阅

cc(1)、 stdsyms(5)。

名称

stat： stat.h - stat() 函数返回的数据

概要

#include <sys/stat.h>

说明

头文件 **<sys/stat.h>** 定义了由函数 **fstat()** 、 **lstat()** 和 **stat()** 返回的数据结构。 **stat** 结构至少包含了以下成员：

dev_t	**st_dev**	包含文件的设备 ID
ino_t	**st_ino**	文件序号
mode_t	**st_mode**	文件模式（请参阅下文）
nlink_t	**st_nlink**	与文件的链接数
uid_t	**st_uid**	文件的用户 ID
gid_t	**st_gid**	文件的组 ID
dev_t	**st_rdev**	设备 ID（如果文件为字符专用设备或块专用设备）
off_t	**st_size**	以字节为单位的文件大小（如果文件为常规文件）
time_t	**st_atime**	上次访问的时间
time_t	**st_mtime**	上次数据修改的时间
time_t	**st_ctime**	上次状态更改的时间
long	**st_blksize**	用于该对象的特定于文件系统的首选 I/O 块大小。在某些文件系统类型中，它可能随文件不同而变化
blkcnt_t	**st_blocks**	分配给该对象的特定于文件系统的大小的块数量
short	**st_fstype**	该文件所在的文件系统的类型；请参阅 *vfsmount*(2)
dev_t	**st_realdev**	包含该文件的 i 节点的设备的实际设备编号

同时使用文件序号和设备 ID 对系统内的文件进行唯一标识。对 **dev_t** 、 **ino_t** 、 **mode_t** 、 **nlink_t** 、 **uid_t** 、 **gid_t** 、 **off_t** 、 **time_t** 和 **blkcnt_t** 等类型进行了定义，如 **<sys/types.h>** 中所述。自历元起，时间按秒给定。

下面还定义了用于 **st_mode** 值的符号名：

文件类型

S_IFMT	**0170000**	文件的类型
S_IFSOCK	**0140000**	套接字
S_IFLNK	**0120000**	符号链接
S_IFNWK	**0110000**	网络设备专用
S_IFREG	**0100000**	常规（普通）
S_IFBLK	**0060000**	块设备专用
S_IFDIR	**0040000**	目录
S_IFCHR	**0020000**	字符设备专用

S_IFIFO	**0010000**	FIFO 设备专用（命名管道）

文件模式位：其他

S_CDF	**0004000**	目录是上下文相关的文件
S_ISUID	**0004000**	执行中设置用户 ID
S_ISGID	**0002000**	执行中设置组 ID
S_ENFMT	**0002000**	设置文件锁定模式为强制锁定
S_ISVTX	**0001000**	在目录文件上设置粘着位

文件模式位：权限

S_IRWXU	**0000700**	所有者的文件访问权限位
S_IRUSR	**0000400**	所有者的读取访问权限
S_IWUSR	**0000200**	所有者的写入访问权限
S_IXUSR	**0000100**	所有者的执行/搜索访问权限
S_IRWXG	**0000070**	组的文件访问权限位
S_IRGRP	**0000040**	组的读取访问权限
S_IWGRP	**0000020**	组的写入访问权限
S_IXGRP	**0000010**	组的执行/搜索访问权限
S_IRWXO	**0000007**	其他用户的访问权限位
S_IROTH	**0000004**	其他用户的读取访问权限
S_IWOTH	**0000002**	其他用户的写入访问权限
S_IXOTH	**0000001**	其他用户的执行/搜索访问权限

文件模式位：过时权限名称

S_IREAD	**0000400**	所有者的读取访问权限
S_IWRITE	**0000200**	所有者的写入访问权限
S_IEXEC	**0000100**	所有者的执行/搜索访问权限

由 **S_IRUSR** 、 **S_IWUSR** 、 **S_IXUSR** 、 **S_IRGRP** 、 **S_IWGRP** 、 **S_IXGRP** 、 **S_IROTH** 、 **S_IWOTH** 、 **S_IXOTH** 、 **S_ISUID** 、 **S_ISGID** 和 **S_ISVTX** 定义的位是唯一的。 **S_IRWXU** 是 **S_IRUSR** 、 **S_IWUSR** 和 **S_IXUSR** 的按位或运算值。 **S_IRWXG** 是 **S_IRGRP** 、 **S_IWGRP** 和 **S_IXGRP** 的按位或运算值。 **S_IRWXO** 是 **S_IROTH** 、 **S_IWOTH** 和 **S_IXOTH** 的按位或运算值。

实现可与其他实现相关的位或运算为 **S_IRWXU** 、 **S_IRWXG** 和 **S_IRWXO** ，但是它们不会与该文档中定义的其他任何位发生重叠。文件权限位定义为与 **S_IRWXU** 、 **S_IRWXG** 和 **S_IRWXO** 的按位或运算值对应的那些位值。

下列宏将测试文件是否为指定类型。提供给宏的值 *m* 是 **stat** 结构中的 **st_mode** 的值。如果测试为 True，则宏计算为一个非零值；如果测试为 False，则宏为 0。

S_ISBLK(*m*) 用于测试块设备专用文件。

S_ISCHR(*m*) 用于测试字符设备专用文件。
S_ISDIR(*m*) 用于测试目录。
S_ISFIFO(*m*) 用于测试管道或 FIFO 设备专用文件。
S_ISREG(*m*) 用于测试常规文件。
S_ISLNK(*m*) 用于测试符号链接。

S_ISCDF(*m*) 用于测试上下文相关文件
S_ISNWK(*m*) 用于测试网络设备专用文件
S_ISSOCK(*m*) 用于测试套接字

以下内容将声明为函数，也可定义为宏：

```
int     chmod(const char *path, mode_t mode);
int     lstat(const char *path, struct stat *buf);
int     mkdir(const char *path, mode_t mode);
int     mkfifo(const char *path, mode_t mode);
int     mknod(const char *path, mode_t mode, dev_t dev);
int     stat(const char *path, struct stat *buf);
mode_t  umask(mode_t cmask);
```

建议使用宏确定文件类型。

警告

对于 32 位应用程序，在使用 64 位值的文件系统中， **st_ino** 将被截断成最低 32 位有效位形式。

另请参阅

chmod(2)、 chown(2)、 link(2)、 mkdir(2)、 mkfifo(3C)、 mknod(2)、 stat(2)、 symlink(2)、 umask(2)、 utime(2)、 types(5)。

符合的标准

<sys/stat.h>：AES、SVID2、SVID3、XPG2、XPG3、XPG4、FIPS 151-2、POSIX.1

历史变更记录

在第 1 期中首次发布。

源自 SVID 第 1 期。

第 4 期

将包含下列更改，以便与 ISO POSIX-1 标准保持一致：

- 该头文件中的函数声明将扩展为完整的 ISO C 原型。
- “说明”一节的内容已扩展，其中指明 (a) 在系统内如何唯一标识文件； (b) 自历元起，时间以秒为单位给定； (c) 控制文件模式位定义和使用的规则； (d) 文件类型测试宏的使用方法。

其他更改合并如下所示：

- 添加对头文件 **<sys/types.h>** 的引用，用于定义 **dev_t** 、 **ino_t** 、 **mode_t** 、 **nlink_t** 、 **uid_t** 、 **gid_t**、 **off_t** 和 **time_t** 。这已标记为一个扩展。
- 已删除对 **S_IREAD** 、 **S_IWRITE** 、 **S_IEXEC** 文件和 **S_ISVTX** 模式的引用。
- 更正了“说明”一节中对 **stat** 结构的成员的描述。

第 **4** 期，第 **2** 版

下列更改将合并，以便与 X/OPEN UNIX 保持一致：

- 将 **st_blksize** 和 **st_blocks** 成员添加到 **stat** 结构中。
- 定义了 **S_IFMT** 的 **S_IFLINK** 值。
- 定义了 **S_ISVTX** 文件模式位与 **S_ISLNK** 文件类型测试宏。
- 将 **fchmod()** 、 **lstat()** 和 **mknod()** 函数添加到该头文件中声明的函数列表中。

名称

st_ats_enabled - 确定打开时是否保留磁带设备

值

保证安全

0（禁用）

缺省值

1（启用）

允许值

0 （禁用）或 任意正整数（启用）

建议值

0 （禁用）或 **1** （启用）

说明

此可调参数通知 stape 或 estape 驱动程序在打开时是否需要保留磁带设备，并在关闭后释放该设备。通过可调参数 **on**， stape/estape 驱动程序不必在打开时尝试保留任何磁带设备。该驱动程序有多个磁带设备标记为适合于此功能的设备。此列表包括 DLT 8000、DLT 7000、STK 9840、HP Ultrium 和 STK SD-3 驱动器。 “未将 *DDS* 驱动器标记为适合此功能的设备。”

通过可调参数 **st_ats_enabled** ，磁带设备可在多节点配置下安全共享，类似于 MC/ServiceGuard 的高级磁带共享。自动保留和释放保护磁带设备不会被可能会损坏备份的多个节点访问。如果 stape/estape 驱动程序的打开例行程序的保留部分失败，则返回 [EBUSY] 状态。

更改此可调参数的人员

任何用户。

更改限制

对此可调参数的更改将在下次重新引导时生效。

应该何时启用此可调参数选项

如果系统要用于 MC/SG 高级磁带共享配置，或用户需要 stape/estape 驱动程序在打开和关闭时对多模式管理使用自动保留/释放，则该可调参数状态应设置为 **on** 。

启用此可调参数的负面影响

stape/estape 驱动程序在打开时将向磁带设备发送一条保留信息，关闭时发送一条释放信息。如果在另一个启动程序（或 HBA）上打开磁带设备，则其他任何启动程序（或 HBA）都将无法对其进行访问。

应该何时禁用此可调参数选项

建议对于除 ATS 以外的任何 SAN 磁带共享解决方案配置，都应将此可调参数设为 **off**。类似 Omniback 的大多数多节点备份应用程序自己管理设备的保留，而 stape/estape 驱动程序的任何介入都可能产生问题。对于任何多平台 SAN 配置，也应将此可调参数设为 **off** ，以使跨平台的磁带访问更加一致。

禁用此可调参数的负面影响

通过其他启动程序进行的未授权访问可能干预任何当前的磁带操作。

应该同时更改的其他可调参数

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

st_ats_enabled 由 HP 开发。

另请参阅

scsi_tape(7)。

名称

stdarg.h - 用于处理变量参数列表的宏

概要

#include <stdarg.h>

void va_start(va_list pvar, argN);
type **va_arg(va_list pvar,** *type***);**
void va_end(va_list pvar);

说明

头文件 **<stdarg.h>** 包含了一组可用于编写接受变量参数列表的可移植过程的宏。包含变量参数列表（例如 **printf()** ），但不使用 *stdarg* 的例行程序，由于不同计算机使用不同的参数传递惯例，因此它们实际上是不可移植的。

va_list 是定义的用于遍历列表的变量的类型。

调用 **va_start** 来初始化 *pvar* 到列表的起始位置。 *argN* 的类型应当与恰好位于参数列表的变量部分之前的函数的参数相同。

va_arg 操作返回 *pvar* 指向的列表中的下一个参数。 *type* 是该参数应为的类型。不同的类型可以混用，但是由例行程序确定所需的参数类型，这是因为它在运行时无法确定。

va_end 用于清除。

可能可以进行多次遍历（每次用 **va_start** ... **va_end** 括起来）。

注释：头文件 **<stdarg.h>** 将替换头文件 **<varargs.h>** ，并且包含所有的 **varargs** 宏。 **<varargs.h>** 提供用于与 ANSI 前的编译器和较早版本的 HP C/HP-UX 兼容。

举例

本示例是 **execl** 的一种可能的实现（请参阅 *exec*(2) ）：

```
#include <stdarg.h>
#define MAXARGS 100

/*  execl is called by
    execl(file, arg1, arg2, ..., (char *)0);
*/
execl(const char *file, const char *args, ...)
{
    va_list ap;
    char *array[MAXARGS];
    int argno = 0;

    va_start(ap, args);
```

```
        if ((array[0] = args) != 0)
            while ((array[argno++] = va_arg(ap, char *)) != 0)
                ;
        va_end(ap);
        return execv(file, array);
    }
```

警告

调用例行程序可指定参数的数量，因为不是总可以通过堆栈帧来确定这一数量的。例如，向 **execl()** 传递一个零指针以标识列表结束，并且 **printf()** 可通过格式字符串确定参数的数量。

除非使用 ANSI C，否则指定 *char* 、 *short* 或 *float* 的第二个参数到 *va_arg* 是不可移植的，这是因为调用的函数看到的这些参数决不是 *char* 、 *short* 或 *float* 。

在将这些参数传递给函数之前， 会将 *char* 和 *short* 参数转换为 *int* ，并且将 *float* 参数转换为 *double* 。

另请参阅

exec(2)、 vprintf(3S)、 varargs(5)。

符合的标准

<stdarg.h>： AES、 SVID3、 XPG4、 FIPS 151-2、 POSIX.1、 ANSI C

va_arg： SVID3、 XPG4、 ANSI C

va_end： SVID3、 XPG4、 ANSI C

va_list： SVID3、 XPG4、 ANSI C

va_start： SVID3、 XPG4、 ANSI C

名称

stdint - 整型

概要

#include <stdint.h>

说明

此头文件定义具有指定宽度的整型集和对应的宏集。另外，它定义了多个宏，这些宏指定与其他标准头文件中所定义的类型相对应的整型的限制。

由于本手册页中定义的所有整数大小不需要所有的实现支持，因此了解当前实现是否支持一个整数的特定大小的正确方式是，测试定义其最大值的符号。例如，如果 **#ifdef UINT64_MAX** 测试失败，那么此实现就不支持 64 位无符号整数。

此头文件定义的 8 位、16 位、32 位和 64 位整型数据类型如下。

intmax_t	实现支持的最大有符号整型数据类型
int8_t	8 位有符号整数
int16_t	16 位有符号整数
int32_t	32 位有符号整数
int64_t	64 位有符号整数
uintmax_t	实现支持的最大无符号整型数据类型
uint8_t	8 位无符号整数
uint16_t	16 位无符号整数
uint32_t	32 位无符号整数
uint64_t	64 位无符号整数

下列两个数据类型是有符号和无符号整型数据类型，此两种类型非常大，足以支持指针。可将指针移至或移离这些数据类型，而不会导致任何损坏。

intptr_t	大得足以保存指针的有符号整型
uintptr_t	大得足够保存指针的无符号整型

为了确定对某个特定实现上的整数值最有效的数据类型，此头文件定义了以下整型数据类型。

intfast_t	实现支持的最高效的有符号整型数据类型
int_fast8_t	最高效有符号整数至少为 8 位

int_fast16_t	最高效有符号整数至少为 16 位
int_fast32_t	最高效有符号整数至少为 32 位
int_fast64_t	最高效有符号整数至少为 64 位
uintfast_t	实现支持的最高效无符号整型数据类型
uint_fast8_t	最高效无符号整数至少为 8 位
uint_fast16_t	最高效无符号整数至少为 16 位
uint_fast32_t	最高效无符号整数至少为 32 位
uint_fast64_t	最高效无符号整数至少为 64 位

为了与不适合 16 位或 32 位字大小模型的系统相兼容，此头文件定义了以下整型数据类型。这些数据类型定义了至少 8 位、16 位、32 位和 64 位的有符号和无符号整数。

int_least8_t	最小有符号整数至少为 8 位
int_least16_t	最小有符号整数至少为 16 位
int_least32_t	最小有符号整数至少为 32 位
int_least64_t	最小有符号整数至少为 64 位
uint_least8_t	最小无符号整数至少为 8 位
uint_least16_t	最小无符号整数至少为 16 位
uint_least32_t	最小无符号整数至少为 32 位
uint_least64_t	最小无符号整数至少为 64 位

下列宏定义了可以存储于上述数据类型中的最小值和最大值。

INTMAX_MIN	可存储于最大整型数据类型中的最小值
INTMAX_MAX	可存储于最大有符号整型数据类型中的最大值
UINTMAX_MAX	可存储于最大无符号整型数据类型中的最大值
INTFAST_MIN	可存储于最高效整型数据类型中的最小值
INTFAST_MAX	可存储于最高效有符号整型数据类型中的最大值
UINTFAST_MAX	可存储于最高效无符号整型数据类型中的最大值
INT8_MIN	可存储于 **int8_t** 数据类型中的最小值

INT16_MIN 可存储于 **int16_t** 数据类型中的最小值

INT32_MIN 可存储于 **int32_t** 数据类型中的最小值

INT64_MIN 可存储于 **int64_t** 数据类型中的最小值

INT8_MAX 可存储于 **int8_t** 数据类型中的最大值

INT16_MAX 可存储于 **int16_t** 数据类型中的最大值

INT32_MAX 可存储于 **int32_t** 数据类型中的最大值

INT64_MAX 可存储于 **int64_t** 数据类型中的最大值

UINT8_MAX 可存储于 **uint8_t** 数据类型中的最大值

UINT16_MAX 可存储于 **uint16_t** 数据类型中的最大值

UINT32_MAX 可存储于 **uint32_t** 数据类型中的最大值

UINT64_MAX 可存储于 **uint64_t** 数据类型中的最大值

INT_FAST8_MIN 可存储于 **int_fast8_t** 数据类型中的最小值

INT_FAST16_MIN 可存储于 **int_fast16_t** 数据类型中的最小值

INT_FAST32_MIN 可存储于 **int_fast32_t** 数据类型中的最小值

INT_FAST64_MIN 可存储于 **int_fast64_t** 数据类型中的最小值

INT_FAST8_MAX 可存储于 **int_fast8_t** 数据类型中的最大值

INT_FAST16_MAX 可存储于 **int_fast16_t** 数据类型中的最大值

INT_FAST32_MAX 可存储于 **int_fast32_t** 数据类型中的最大值

INT_FAST64_MAX 可存储于 **int_fast64_t** 数据类型中的最大值

INT_LEAST8_MIN 可存储于 **int_least8_t** 数据类型中的最小值

INT_LEAST16_MIN 可存储于 **int_least16_t** 数据类型中的最小值

INT_LEAST32_MIN 可存储于 **int_least_32_t** 数据类型中的最小值

INT_LEAST64_MIN 可存储于 **int_least_64_t** 数据类型中的最小值

INT_LEAST8_MAX 可存储于 **int_least8_t** 数据类型中的最大值

INT_LEAST16_MAX 可存储于 **int_least16_t** 数据类型中的最大值

INT_LEAST32_MAX 可存储于 **int_least_32_t** 数据类型中的最大值

INT_LEAST64_MAX 可存储于 **int_least_64_t** 数据类型中的最大值

下列宏指定与其他标准头所定义类型对应的整数类型的最大和最小限值。所有这些值均由实现定义。

PTRDIFF_MIN 可存储于 **ptrdiff_t** 数据类型中的最小值

PTRDIFF_MAX 可存储于 **ptrdiff_t** 数据类型中的最大值

SIG_ATOMIC_MIN 可存储于 **sig_atomic_t** 数据类型中的最小值

SIG_ATOMIC_MAX 可存储于 **sig_atomic_t** 数据类型中的最大值

SIZE_MAX 可存储于 **size_t** 数据类型中的最大值

WCHAR_MIN 可存储于 **wchar_t** 数据类型中的最小值

WCHAR_MAX 可存储于 **wchar_t** 数据类型中的最大值

WINT_MIN 可存储于 **wint_t** 数据类型中的最小值

WINT_MAX 可存储于 **wint_t** 数据类型中的最大值

下列宏扩展为适用于初始化对象的整数常量表达式，这些对象具有与 **<stdint.h>** 头文件所定义类型对应的整型。

最小宽度的整数常量表达式的宏

宏 **INTN_C(***value***)** 扩展为与 **int_leastN_t** 类型对应的整数常量表达式。

宏 **UINTN_C(***value***)** 扩展为与 **uint_leastN_t** 类型对应的整数常量表达式。例如，如果 **uint_least64_t** 是 **unsigned long long** 类型的名称，则 **UINT64_C(0x123)** 可扩展为 **0x123ULL** 整数常量。

最大宽度的整数常量表达式的宏

下列宏扩展为具有其参数和 **intmax_t:INTMAX_C(***value***)** 类型所指定的值的整数常量表达式

下列宏扩展为具有其参数和 **uintmax_t:UINTMAX_C(***value***)** 类型所指定的值的整数常量表达式。

文件

/usr/include/stdint.h

另请参阅

inttypes(5)、 standards(5)、 <stddef.h>、 <wchar.h>、 <signal.h>。

名称

stdsyms - HP-UX 头文件中命名定义和其他命名空间规范的说明

说明

stdsyms 是关于“命名定义”和其他规范的说明，改说明不许由应用程序设置，用于从 HP-UX 头文件中获得相应的命名空间。

HP-UX 头文件是按如下方式构成的：仅包括头文件可使用的符号子集，以便对于符合特定标准的应用程序是可见的。 ANSI-C、POSIX.1、POSIX.2、XPG4 及 ANSI-C/POSIX/XPG 的后续增强版本各自保留了一个特定的符号集，用于对应标准的的命名空间。此外，XPG3 和 OSF AES/OS 的 HP-UX 实现提供了一个洁净的命名空间，尽管那些标准没有对此进行特定要求。

下面的规则用于确定为任何一个标准保留的符号。这些符号保留用于标准和供实现使用，并且必须完全取消或完全按指定标准的定义使用。

- 所需标准定义的所有符号都将保留。有关保留符号的完整列表，请参考相应的标准文档。
- 所有以下划线开头，后接另一条下划线或一个大写字母的符号都将保留用于实现。
- 所有以下划线开头的外部标识符都将保留用于实现。

以下是一个功能测试宏的列表，这些宏必须进行定义，以从头文件中获得相应的命名空间。

__STDC__
此符号由 ANSI-C 预处理器自动定义，并且在指定 ANSI-C 编译 (**cc -Aa**) 时也将自动定义。使用精确的 ANSI 选项 **-Aa** 可请求获取纯 ANSI-C 命名空间，它是可用的 HP-UX 命名空间的最小子集。 **-Aa** 选项也可包括 ANSI-C 样式的函数原型，用于进行增强的类型检查。请注意，当使用 **-Aa** 选项时，缺省的命名空间为 ANSI-C 命名空间；因此如果需要，必须要求更大的命名空间。

__STDC_VERSION__
在某些实现中，此符号由 ANSI-C 预处理器自动定义。如果已定义，该符号则表示其符合的标准版本。

_POSIX_SOURCE
正如 IEEE POSIX.1 标准中记录的，程序员需要对 **_POSIX_SOURCE** 功能测试宏进行定义，以获得 POSIX.1 命名空间和 POSIX.1 功能。此功能测试宏可通过下列方法定义：使用编译器选项 (**-D_POSIX_SOURCE**) ，或者在所有 **#include** 指令之前使用源文件中的 **#define** 指令。请注意，缺省的 POSIX 命名空间是 POSIX.1-1990 命名空间。为了获得 POSIX.1-1988 命名空间，除了需要 **_POSIX_SOURCE** 宏外，还必须定义 **_POSIX1_1988** 功能测试宏。

_POSIX_C_SOURCE
正如 IEEE POSIX.2 标准中记录的，程序员需要对值为 2 的 **_POSIX_C_SOURCE** 功能测试宏进行定义，以获得 POSIX.1 及 POSIX.2 命名空间和功能。此功能测试宏可通过下列方法定义：使用编译器选项 (**-D_POSIX_C_SOURCE=2**) ，或者在所有 **#include** 指令之前使用源文件中的 **#define** 指令。

_XOPEN_SOURCE
正如《X/Open Portability Guide》 (**XPG**) 中记录的，程序员需要对 **_XOPEN_SOURCE** 功能测试宏进行定义，以获得 X/Open 功能。此功能测试宏可通

过下列方法定义：使用编译器选项 (**-D_XOPEN_SOURCE**) ，或者在所有 **#include** 指令之前使用源文件中的 **#define** 指令。尽管 XPG3 未指定任何命名空间污染规则，但是 XPG4 及其后续版本已制定了这样的规则。因此，只要定义了 **_XOPEN_SOURCE**，HP-UX 操作系统就会提供洁净的命名空间。

当前缺省的 X/Open 命名空间是对应于 XPG4 的那个命名空间。通过将 **_XOPEN_SOURCE** 设置为标准文档中指定的值，可以请求更大的命名空间。

_XOPEN_SOURCE_EXTENDED

按照 XPG 中的规定，程序员需要对 **_XOPEN_SOURCE_EXTENDED** 功能测试宏进行定义，以获得 **XPG4 v2** 命名空间和功能。此功能测试宏可通过下列方法定义：使用编译器选项 (**-D_XOPEN_SOURCE_EXTENDED**) ，或者在所有 **#include** 指令之前使用源文件中的 **#define** 指令。

_AES_SOURCE

按照“OSF AES/OS”标准中的规定，程序员需要对 **_AES_SOURCE** 功能测试宏进行定义，以获得 OSF 功能。此功能测试宏可通过下列方法定义：使用编译器选项 (**-D_AES_SOURCE**) ，或者在所有 **#include** 指令之前使用源文件中的 **#define** 指令。尽管 AES 未指定任何命名空间污染规则，但是其他标准已制定了这样的规则。因此，只要定义了 **_AES_SOURCE**， HP-UX 就会提供洁净的命名空间。强烈建议不要使用 **_AES_SOURCE**，因为在 HP-UX 的将来版本中， 将删除此功能。

_HPUX_SOURCE

程序员可以定义 **_HPUX_SOURCE** 功能测试宏，以获得 HP-UX 命名空间和全部的 HP-UX 功能。请注意，HP-UX 命名空间当前是上述所有命名空间的超集。当使用带有缺省选项 (**cc**) 的编译器，或使用带有兼容模式选项的编译器 （**cc** 命令，不带 **-Aa** 选项）时，缺省情况下将提供 HP-UX 命名空间（请参阅 *cc*(1)）。程序员必须请求上述其他命名空间中的一个，以获得 HP-UX 命名空间的相应子集。当使用精确的 ANSI-C-mode 编译器 (**cc -Aa**) 时，程序员必须特别要求一个更大的命名空间。

_HPUX_SOURCE 功能测试宏可通过下列方法定义：使用编译器选项 (**-D_HPUX_SOURCE**)，或者在所有 **#include** 指令使用源文件中的 **#define** 指令。

以下为其他功能测试宏的列表，用于提供各种不同的附加功能。

__cplusplus

该符号由 HP C++ 编译器自动定义。对此宏进行定义可在系统头文件中启用 C++ 函数原型。

HP C++ 的缺省命名空间是 ANSI-C 命名空间。要获得其他命名空间，请定义相应的功能测试宏。

缺省情况下，HP C++ 将使用 ANSI-C 预处理器。要获得兼容模式的预处理器，请使用 **cc** 命令的 **-Ac** 选项（请参阅 *cc*(1)）。兼容模式的预处理器使用 HP-UX 命名空间 (**_HPUX_SOURCE**)。

_POSIX1_1988

当需要 **POSIX.1-1988** 命名空间时，应对此功能测试宏进行定义。如果不需要缺省的 **POSIX.1-1990** 命名空间，则此宏应与 **_POSIX_SOURCE** 宏一起使用。

每当请求获取 **_AES_SOURCE** 或 **_XPG3** 时，将自动对此宏进行定义。

_XPG3
由于 XPG3 命名空间与 **_XPG4** 命名空间稍有不同，因此提供了 **_XPG3** 功能测试宏，以便程序员可以获得 XPG3 命名空间。为了获得 XPG3 命名空间，程序员必须定义 **_XOPEN_SOURCE** 和 **_XPG3** 两个功能测试宏。 **_XOPEN_SOURCE** 和 **_XPG3** 功能测试宏可通过方法定义：使用编译器选项 (**-D_XOPEN_SOURCE -D_XPG3**) 定义，或者在所有 **#include** 指令之前使用源文件中的 **#define** 指令。强烈建议不要使用此宏，因为在 HP-UX 的将来版本中，将删除此功能。

_XPG4
如果程序员请求获取 XPG4 命名空间，则 **_XPG4** 功能测试宏将自动定义（即，定义了 **_XOPEN_SOURCE** ，而不是其他一些有冲突的命名空间，如 **_XPG3** ）。

_SVID2
当使用兼容模式的编译器时，可以定义 **_SVID2** 宏，用于在 HP-UX 命名空间中获得 SVID2 函数返回类型。在 HP-UX 操作系统中，许多函数的缺省返回类型后来已更改，以与 ANSI-C、POSIX、X/Open 和 OSF 等标准保持一致。强烈建议不要使用此宏，因为在 HP-UX 的将来版本中，将删除此功能。

_SVID3
可以定义 SVID3 宏以获得 SVID3 函数原型。需要定义一个编译器标记 **-D_SVID3** ，用于表示为符合 SVID3 的要求，已写入应用程序。当 ANSI C 中引入函数原型时，此标记引出的函数仅在 SVID3 中定义。

另请参阅

cc(1)、 cpp(1)、 pathconf(2)、 sysconf(2)、 standards(5)。

名称

STRCTLSZ - 流消息控制最大大小（字节）

值

无故障

1024

缺省值

1024

允许值

0 - 2147483647

建议值

1024

说明

STRCTLSZ 限制控制数据的最大字节数，控制数据可以通过 **putmsg()** 操作插入到系统中任何数据流消息的控制部分中。如果此可调参数设置为零，则对能够放入消息控制段的字节数没有任何限制。

如果发送的缓冲区大于 **STRCTLSZ** 的当前值，则 **putmsg()** 将返回 [ERANGE] 。

应该由谁来更改此可调参数？

任何客户。

对于更改的限制

对此可调参数的更改将在下次重新引导时生效。

何时应增加此可调参数的值？

如果客户的 STREAMS 模块/驱动程序在任意流消息的控制部分中需要比当前值更多的字节，则任何客户都应该增加此可调参数的值。

增加此可调参数值的副作用是什么？

内核将使用更多的内存。在低内存条件下，可能会因频繁交换而导致导致系统性能下降。

何时应降低此可调参数的值？

如果客户的 STREAMS 模块/驱动程序对于控制部分不需要比当前值更长的消息长度，则任何客户都可以降低这一可调参数的值。

降低此可调参数值的副作用是什么？

可能会导致一些 STREAMS 模块/驱动程序运行错误。可能导致性能下降，尤其是网络方面。

同时还应更改哪些其他可调参数的值？

无。

警告

所有 HP-UX 内核可调参数都是针对于特定版本的。将来的 HP-UX 版本中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅网站 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

STRCTLSZ 由 HP 开发。

名称

streampipes - 强制所有管道为基于 STREAMS 的管道

值

无故障

0

缺省值

0

允许值

0 - 2147483647

建议值

0

说明

此可调参数可确定 **pipe()** 系统调用创建的管道类型。如果将缺省值设定为零，则 **pipe()** 创建的所有管道都是普通 HP-UX 文件系统管道。如果值不为零，则 **pipe()** 将创建基于 STREAMS 的管道，并且可以将 STREAMS 模块置入生成的流中。

如果此可调参数设置为非零值，则 **pipemod** 和 **pipedev** 模块和驱动程序必须在文件 **/stand/system** 中配置。

应该由谁来更改此可调参数？

任何客户。

对于更改的限制

对此可调参数的更改将在下次重新引导时生效。

何时应启用此可调参数选项？

如果客户使用需要基于 STREAMS 的管道的应用程序，则应启用此可调参数。

启用此可调参数的副作用是什么？

基于 STREAMS 的管道性能可能与普通文件系统管道有所不同。

何时应禁用此可调参数选项？

如果客户不需要基于 STREAMS 的管道，则应禁用可调参数。

禁用此可调参数的副作用是什么？

尝试将 STREAMS 模块置入管道的应用程序将失败。

同时还应更改哪些其他可调参数的值？

如果此可调参数设置为非零值，则必须在文件 **/stand/system** 中配置 **pipemod** 和 **pipedev** 模块和驱动程序。

警告

所有 HP-UX 内核可调参数都是针对于特定版本的。将来的 HP-UX 版本中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺

省值或建议的值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅网站 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

streampipes 由 HP 开发。

名称

STRMSGSZ - 流消息数据最大大小（字节）

值

无故障

0

缺省值

0

允许值

0 - 2147483647

建议值

0

说明

此可调参数可限制消息数据的最大字节数，消息数据可以通过 **putmsg()** 或 **write()** 操作插入到系统中任何数据流消息的数据部分中。 如果此可调参数设置为零，则对能够放入消息数据段的字节数没有任何限制。

如果发送的缓冲区大于 **STRMSGSZ** 的当前值，则 **putmsg()** 将返回 [ERANGE]； **write()** 可将数据划分到多条消息中。

应该由谁来更改此可调参数？

任何客户。

对于更改的限制

对此可调参数的更改将在下次重新引导时生效。

何时应增加此可调参数的值？

如果客户的 STREAMS 模块/驱动程序对于数据部分需要比当前值更长的消息长度，则任何客户都应该增加这一可调参数的值。

增加此可调参数值的副作用是什么？

内核将使用更多的内存。在低内存条件下，可能会因频繁交换而导致导致系统性能下降。

何时应降低此可调参数的值？

如果客户的 STREAMS 模块/驱动程序对于数据部分不需要比当前值更长的消息长度，则任何客户都应该降低这一可调参数的值。

降低此可调参数值的副作用是什么？

可能会导致一些 STREAMS 模块/驱动程序运行错误。可能导致性能下降，尤其是网络方面。

同时还应更改哪些其他可调参数的值？

无。

警告

所有 HP-UX 内核可调参数都是针对于特定版本的。将来的 HP-UX 版本中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅网站 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

STRMSGSZ 由 HP 开发。

名称

suffix - 文件名后缀约定

说明

以下列表总结了 HP-UX 系统中包含的文件名后缀的约定。其中仅编辑收录了可能有用的部分知识、建议和说明，而不是一份标准规范。通常优先选择使用后缀而不是前缀，因为后缀使得相关的文件能够按字母顺序在目录列表中组合。

请注意，某些程序需要使用特定值，或基于选择的后缀改变其行为。在很多（不是所有）情况下都对这类程序进行了注释。

.A HP64000 交叉汇编程序符号文件。

.a **ar** 管理的库文件（归档）；适用于 **make** 。

.ad? HP Ada 源，其中“?”代表任意单个字符。

.allow **at** 或 **cron** 允许使用的用户列表（例如， **at.allow** ）。

.an **nroff man** 宏的源。

.ASC LIF （逻辑交换格式）类型 1，供 Pascal 或 BASIX/UX 使用的 ASCII 文件。与 **lifcp** 不兼容。

.aux LaTeX 自动创建的交叉引用信息。

.awk **awk** 脚本文件。

.b 已编译的 (**.l**) 源文件，或粗体文件。

.back, **.bak**, **.bkup**
文件的备份副本。

.BAD, **.bad** 包含无效数据或占用磁盘无效空间的文件。

.bbl 由 BibTeX 创建的包括在 LaTeX 文档中的参考书目。

.bib 参考书目数据文件，（例如，BibTeX 参考书目数据库）。

.blg BibTex 中的错误日志。

.bst BibTeX 参考书目样式定义。

.B **compact** 压缩的文件或 C++ 语言源文件，或 HP64000 交叉编译的 C 源文件。

.c C 语言源文件；适用于 **cc** 和 **make** 。

.cas CAST 语言脚本。

.cat NLS（本地语言支持）消息清单。

.cf 配置文件（例如， **sendmail.cf** ）。

.clu	CLU 文件。
.BODE	Pascal 工作站对象代码。
.cpio	包含从 **cpio -o** 输出的文件，即 **cpio** 归档。
.csh	C Shell (**csh**) 脚本。
.curr	文件的当前版本。
.d	目录文件，或数据文件。
.day	每日读取的脚本。
.deny	**at** 或 **cron** 拒绝的用户列表（例如， **cron.deny** ）。
.devs	设备列表。
.diff	**diff** 输出的两个文件的差异。
.dir	DBM 数据库目录文件。
.doc	部分排序的文档文件。
.dvi	设备独立的文本格式设置程序输出。
.e	扩展的 FORTRAN 语言 (EFL) 源文件；适用于 **make**
.el	GNU Emacs Elisp 文件。
.elc	已编译的 GNU Emacs Elisp 文件。
.eqn	**nroff** 方程式宏的源。
.err	来自程序的标准错误。
.errors, **.errs**	程序记录的错误。
.f	FORTRAN 语言源文件；适用于 **fc** 和 **make**
.f77	FORTRAN 77 语言源文件。
.fc	冻结的配置文件（例如， **sendmail.fc** ）。
.full	一个完整的文件或列表。
.gf	格式为 Generic Font 的 TeX 字体位图。
.glo	LaTeX 创建的词汇表。
.h	C 语言头（包含）文件；适用于 **make** 。
.help, **.hf**, **.hlp**	程序的帮助文本，通常会自动读取。

.hour, **.hr** 每小时读取的脚本。

.i C 预处理器 (**cc -P**), 的输出，或 Berkeley Pascal 语言包含文件，或斜体字文件。

.icn 图标源代码。

.idx LaTeX 创建的索引。

.in 标准程序输入。

.INDEX Notes 索引文件。

.ksh Korn shell 脚本文件。

.L HP64000 交叉链接程序符号文件。

.l **lex** 源文件（适用于 **make** 或 LISP 源文件。

.LIST Notes 列表文件。

.list 包含其他文件列表的文件。

.ln **lint** 的库信息。

.lof LaTeX 创建的数字列表。

.log 常规日志文件，或 TeX 中的错误消息日志。

.lot LaTeX 创建的表的列表。

.m 模块语言源文件。

.m2 模块 2 语言源文件。

.make, **.mk** **make** 的生成文件。

.man 使用 **man** 宏的 **nroff** 或 **troff** 的源。

.me 使用 **me** 宏的 **nroff** 或 **troff** 的源。

.mf TeX 元字体输入文件。

.ml Gosling/Unipress Emacs Mock Lisp 文件。

.mm 使用 **mm** 宏的 **nroff** 或 **troff** 的源。

.mon, **.month** 每月读取的脚本。

.ms 使用 **ms** 宏的 **nroff** 或 **troff** 的源。

.n **nroff** 的源。

.NEW, **.new** 文件的新版本。

.nro	**nroff** 的源。
.O	HP64000 列表文件。
.o	可重定位的对象文件（编译后，链接前）；适用于 **as** 、 **cc** 、 **fc** 、 **pc** 和 **make** 。
.obs	文件的过时版本。
.OLD, **.old**	文件的旧版本。
.opt	包含可选内容的文件，如内核的可选部分。
.orig	文件的原始版本。
.out	源自程序的标准输出（也可能是标准错误）（例如， **nohup.out** ），或 **ld** 输出的可执行文件（如 **a.out** ）。
.P	HP64000 交叉编译的 Pascal 源文件。
.p	Pascal 语言源文件（适用于 **pc** 和 **make** ）或 PROLOG 语言源文件。
.pag	DBM 数据库数据文件。
.pi	PILOT 语言源文件。
.pk	格式为 Packed Font 的 TeX 字体位图；较 GF 更密集/更近。
.prev	文件的先前版本。
.ps	PostScript 文件。
.pxl	未压缩格式的 TeX 字体位图；完全过时。
.R	HP64000 可重定位文件。
.r	RatFor 语言源文件；适用于 **make** 。
.rc	“运行命令”文件，通常在调用程序时读取（例如， **mailx.rc** ）。
.real	文件的实际版本，通常被前端替换（例如， **uucico.real** ）。
.req	包含必需内容的文件，如必需的部分内核。
.S	HP64000 交叉汇编的源文件。
.s	汇编程序输入文件；适用于 **cc** 和 **make** 。
.safe, **.save**	文件的安全或保存的副本。
.scm	方案文件。
.sh	POSIX Shell 脚本文件；适用于 **make** 。
.shar	包含 **shar** 输出的 Shell 归档文件。

.skel 主干文件或模板文件。

.sl **ld;** 构建的共享库文件，适用于 **ld** 。

.st 包含统计数据的文件（例如， **/etc/mail/sendmail.st** ）。

.sty LaTeX 样式定义；应该具有相对应的 **.doc** 文件。

.SYSTM LIF 可由 300/400 系列引导 ROM 引导（请参阅《Pascal 3.2 Workstation System》第 1 卷中的“ Librarian”一章）。

.t 文本文件。

.tar 包含 **tar** 输出的文件（归档）。

.tbl **nroff** 表宏的源。

.temp, **.tmp** 临时文件。

.template 原型或模板文件。

.test 测试输入或输出文件。

.tex TeX 源文件。

.TEXT Notes 文本文件，或 Pascal 工作站“UCSD 文本格式”文件。

.text, **.txt** ASCII 文本文件。

.tfm TeX（TeX 字体公制）使用的宽度信息。

.toc 内容宏的 **nroff** 表的源，或 LaTeX 创建的内容表。

.tro **troff** 的源。

.u1, **.u2** 图标中间代码文件。

.UX HP-UX 文本或二进制文件格式。

.web Web 文件（Knuth 的 Web 系统）。

.week, **.wk** 每周读取的脚本。

.X HP64000 绝对文件。

.y **yacc** 输入文件；适用于 **make** 。

.Z **compress** 压缩的文件。

.z **pack** 压缩的文件。

.1 .. **.9** 联机帮助文件（第 1 到 9 节），可以选择后跟字母 **a**..**z** 。HP-UX 还可使用 **.1m** 表示 1M 小节。

.*date* 在指定 日期（例如，年份、月份名称、 *YYMM* 、 *MMDD* 等）保存的文件作为持续增大的日志文件的快照。

,v RCS delta 版文件；适用于 RCS 程序。

作者

suffix 由 HP 开发。

已过时

名称

swapmem_on - 已过时的内核可调参数

说明

swapmem_on 可调参数已过时。将始终允许进程使用伪交换空间（如果此空间可供使用）。

在先前版本的 HP-UX 中，为了得到系统进程的最大可能数量，系统配置需要足够的物理交换空间。这是因为当创建进程时，HP-UX 会为其保留交换空间，以确保运行的进程在任何情况下都不会由于交换空间不足而终止。

然而，这对需要数千兆字节大小的、包含数千兆字节物理内存的系统，以及那些整个工作负荷一直在核心的系统来说，是难以实现的。创建此可调参数可使系统交换空间小于核心内存。为了实现这一点，可保留一部分物理内存作为“伪交换”空间。而实际交换空间仍然可用，进程仍然会在物理设备或文件系统交换派生或执行时保留其所需要的所有空间。一旦完全利用这些空间，新的进程就不能保留交换，同时会改为将每个应已交换到物理设备或文件系统的页面锁定在内存中，并将其计为伪交换空间的一部分。

警告

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

swapmem_on 由 HP 开发。

名称

swchunk - 交换组块大小（以 1KB 块为单位）

值

缺省值

2048 块

允许值

最小值：**2048** 块

最大值： **65536** 块

说明

内核中的交换空间可以使用物理设备空间的“组块”进行管理。这些组块包含内存的一页或多页（通常是多页），但是提供另一层索引（类似于文件系统中的 *inodes* ）来保留相对较小的全局交换表，与通过交换页索引的大型表对应。

swchunk 控制物理磁盘块（定义为 1 KB）中每个组块的大小。系统可管理的交换空间的总字节数为 **swchunk** * 1 KB * 2,147,483,648 （交换表中系统的交换组块的最大数）。请注意 **swchunk** 的最小值（或缺省值）因此可允许 4,096 TB 的交换空间。

swchunk 的推断方式不是交换系统（以磁盘块计）中 I/O 事务的大小，而是在转移到下一个设备（假定所有优先级都相等）前，将放置在一个交换设备（或文件系统）中的块数。这将通过任意设备传输交换空间，称为 *swap* 交错。当交换使用频繁时，交换的交错会将交换通过多个设备输出，并减少单个设备成为整个系统瓶颈的可能性。

更改此可调参数的人员

只有那些对内核行为和基础设备硬件完全了解的人才应修改此可调参数。

更改限制

对此可调参数的更改将在下次重新引导时生效。

应该何时增加此可调参数的值

如果系统所有者希望向系统增加更多的交换，但是需要的附加交换组块不可用，则增加此可调参数将出现问题。通过增大每个组块的大小，需要的总的组块数会更少。

增加此值的负面影响

交换表的第二级（用于跟踪组块中的页）将增大，从而导致内核使用更多的内存。如果为包含更大交换空间的映射而增大 **swchunk** ，则内核为跟踪交换空间，将不可避免增加使用内存。

这表示，当所有优先级都相等时，通过使用循环交错方案，将有更多交换分配给每个设备（或文件系统）。当表示系统交换空间需要的组块数少于 2,147,483,648 时，增大 **swchunk** 会由于创建不必要的 I/O 瓶颈而影响系统性能。例如，在不同组块中使用较小值的两个页面曾经位于不同的交换设备上，因此彼此之间可独立访问（无读取头文件或控制器的问题），现在它们处于同一设备上，并无法同时读取，从而导致对第二页的访问时间更长。

应该何时降低此可调参数的值

如果系统可映射的的交换空间的数量远远大于连接到（或即将连接到）系统的交换空间的总的数量（可通过乘法 2,147,483,648 * **swchunk*** 1KB 计算），则通过降低 **swchunk** 以适合实际交换空间，可减少内核内存的使用。

降低此值的负面影响

如果更多的交换添加到系统中，并且在交换表中没有足够的空间可包含增加的空间，则此值可能必须增大为原值。如果不是这种情况，则在系统中（假定有更多的交换设备）将有较细的交错，可在频繁的交换使用下改进性能。

应该同时更改的其他可调参数值

对 **swchunk** 的更改独立于其他任何可调参数。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

swchunk 由 HP 开发。

名称

sys_attrs_kevm - KEVM（内核事件管理器）子系统属性

说明

此联机帮助页列出和介绍了内核事件管理器 (**kevm**) 子系统的属性。有关事件管理器软件的详细信息，请参阅 *evm*(5) 。

以下列表中，属性名称前带有星号 (*) 表示可以在运行时配置该属性。其他属性值在不重新引导系统的情况下不能更改。

*** do_reset_stats**

该值可以用来重新初始化所有 **kevm stats_*** 属性。将 **do_reset_stats** 更改为任何非零值会将所有 **stats_*** 属性重置为零，然后 **do_reset_stats** 自身将自动重置为零。

缺省值：0

最小值：0

最大值：2,147,483,647

如果使用 **dxkerneltuner** 更改 **do_reset_stats** 的值，则在退出并重新启动 **dxkerneltuner** 之前，更新的属性值将不可见。

events_in_queue_count

当前在内核空间中排队，等待 EVM 守护程序收集的 EVM 事件的数目。该属性只能用于查询。

evm_interrupt_buffer_size

对内核在中断级别执行时发布的任何 EVM 事件进行排队所使用的固定缓冲区的大小（字节）。系统返回正常的操作模式后，事件将立即从缓冲区中移出。缺省的缓冲区大小应该足以满足大多数系统使用。

缺省值：65,536（字节）

最小值：1

最大值：1,844,674,407,370

kevm_is_open

一个值，指示进程（例如 **kevmd** ）当前是否具有处于打开状态的 **kevm** 接口。值 1 表示进程具有打开的接口（如果 **kevmd** 正在运行，则情况始终如此），值 0 表示所有进程都没有打开的接口。该属性只能用于查询。

kevm_major

kevm 接口使用的主设备编号。该属性只能用于查询。

*** stats_events_in**

自启动系统或重置计数器以来，在内核中发布的 EVM 事件的数目。可以直接重置 **stats_events_in** ，也可以将 **do_reset_stats** 更改为非零值。

最小值：0

最大值：2,147,483,647

*** stats_events_read**

自启动系统或重置计数器以来，EVM 守护程序 (**evmd**) 从内核空间读取的 EVM 事件的数目。可以手动重置 **stats_events_read** ，也可以将 **do_reset_stats** 更改为非零值。

最小值：0

最大值：2,147,483,647

*** stats_queue_high_watermark**

自启动系统或重置计数器以来，在内核空间中排队以等待 EVM 守护程序收集的 EVM 事件的最大数目。可以手动重置 **stats_queue_high_watermark** ，也可以将 **do_reset_stats** 更改为非零值。

最小值：0

最大值：2,147,483,647

另请参阅

evm(5)。

系统管理

《EVM System Admnistration Guide》。

名称

sysv_hash_locks - System V IPC 散列 spinlock 池的大小

值

无故障

128

缺省值

128

允许值

最小值： **64**

最大值： **32768** 或 **semmni** 中较小的那个。

需要为 2 的幂

说明

可调参数 **sysv_hash_locks** 指定散列 spinlock （用于同步的内核数据结构）池（可用的 spinlock 的数量）的大小。 System V IPC 信号量函数基于信号量 ID 获取散列 spinlock。将 spinlock 散列是为了避免用户用于 **semmni** spinlock 所需的内存，但也是为了避免争用仅一个 spinlock。

有关 System V 信号量的详细信息，请参考 *mesg*(5) 联机帮助页中的概述一节。

应该由谁来更改此可调参数？

那些通过内核性能分析，已使用这些锁对系统性能问题进行了标识的人员可以更改此参数。 HP 的经验表明很少有客户需要调整此可调参数。

对于更改的限制

对此可调参数的更改将在下次重新引导时生效。

何时应增加此可调参数的值？

如果性能分析显示这些锁被高度争用，则应该增加此可调参数的值。大幅度增加可调参数 **semmni** ，同时伴随着大量使用信号量操作，将可能产生这种情况。

何时应降低此可调参数的值？

如果 **semmni** 不再很大，并且（或者）spinlock 争用不再是问题时，则降低此可调参数的值。

同时还应更改哪些其他可调参数的值？

所有的 System V 信号量可调参数都是相互关联的，而且不应该作为独立变量处理。这些可调参数必须作为一个系统来评估，以保证它们能反映应用程序的要求。这些信号量可调参数包括 **semaem** 、 **semmni** 、 **semmns** 、 **semmnu** 、 **semmsl** 、 **semume** 、 **semvmx** 以及 **sysv_hash_locks** 。尤其是，通过可调参数 **semmni** 对信号量 ID 数量的重大更改可能需要更改可调参数 **sysv_hash_locks** 。

警告

所有 HP-UX 内核可调参数都是针对于特定版本的。将来的 HP-UX 版本中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅网站 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

sysv_hash_locks 由 AT&T 开发。

另请参阅

sema(5)、 semaem(5)、 semmni(5)、 semmns(5)、 semmnu(5)、 semmsl(5)、 semume(5)、 semvmx(5)。

名称

tcphashsz - 确定联网散列表的大小

值

无故障

2048

缺省值

2048

允许值

256 到 **65536**

建议值

2048

指定为 2 的幂或者四舍五入为最接近的 2 的幂。

说明

此变量用于设定联网散列表的大小。将有大量连接接入的系统，任何时候都可以看到增加该值的某种益处。

此可调参数需要设置为 2 的幂。如果未指定为 2 的幂，则应四舍五入到最接近的 *2* 的幂。

应该由谁来更改此可调参数?

任何用户。

对于更改的限制

对此可调参数的更改将在下次重新引导时生效。

何时应增加此可调参数的值?

如果计算机包含大量持续时间较长的连接，则可增加此可调参数的值。

增加此值的副作用是什么?

当增加此参数值时，将会使用更多的内存。

何时应降低此可调参数的值?

如果系统内存紧张，而且只有少量的连接，则降低此可调参数的值可回收部分内存。

降低此值的副作用是什么?

如果此可调参数的值设置过低，则系统将使用很长的散列链，从而导致系统速度下降。

同时还应更改哪些其他可调参数的值?

无。

警告

所有 HP-UX 内核可调参数都是针对于特定版本的。将来的 HP-UX 版本中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上

的可选内核软件的信息，请参阅网站 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

tcphashsz 由 HP 开发。

名称

term - 终端功能

概要

#include <term.h>

说明

下列数据类型通过 **typedef** 定义:

TERMINAL 在 **terminfo** 数据库中单个终端功能的不透明表现形式。

头文件 **<term.h>** 中提供了有关以下对象的声明: *cur_term* 它表示应用程序通过调用 **set_curterm()** 已选择的 **terminfo** 数据库中的当前终端记录。

头文件 **<term.h>** 包含 terminfo(4) 中的表格 **Defined Capabilities** 里的 “**Variable**”列中列出的变量名。

下列内容可声明为函数，也可定义为宏:

```
int        del_curterm(TERMINAL *oterm);
int        putp(char *const str);
int        restartterm(char *term, int fildes, int *errret);
TERMINAL * set_curterm(TERMINAL *nterm);
int        setupterm(char *term, int fildes, int *errret);
int        tgetent(char *bp, char *const name);
int        tgetflag(char id[2]);
int        tgetnum(char id[2]);
char *     tgetstr(char id[2], char **area);
char *     tgoto(char *const cap, int col, int row);
int        tigetflag(char *capname);
int        tigetnum(char *capname);
char *     tigetstr(char *capname);
char *     tparm(char *cap, long p1, long p2, long p3, long p4,
             long p5, long p6, long p7, long p8, long p9);
int        tputs(char *const str, int affcnt, int (*putfunc)(int));
```

另请参阅

terminfo(4)、 printf(1)、 putp(3X)、 tigetflag(3X)、 tgetent(3X)、 <curses.h>。

历史变更记录

在 X/Open Curses 第 4 期中首次发布。

名称

thread_safety - libc、libpthread 和 libgen 接口列表如下：非线程安全、取消点、取消安全、异步信号安全或异步取消安全

说明

当使用 libc、libpthread 和 libgen 接口编写线程安全应用程序时，请参考此联机帮助页。此联机帮助页给出 libc、libpthread 和 libgen 接口的列表，包括：非线程安全、取消点、取消安全、异步信号安全或异步取消安全。

非线程安全的接口

所有接口都是线程安全的，但以下接口除外：

bgets()	**bufsplit()**	**copylist()**
dbmclose()	**dbminit()**	**delete()**
endmntent()	**fattach()**	**fdetach()**
fetch()	**firstkey()**	**getc_unlocked()**
getchar_unlocked()	**getcontext()**	**getopt()**
inet_ntop()	**inet_pton()**	**isastream()**
makecontext()	**memcntl()**	**nextkey()**
putc_unlocked()	**putchar_unlocked()**	**setcontext()**
setmntent()	**store()**	**strord()**
strtold()	**system()**	**swapcontext()**

注释：如果接口有一个 **_r** 对应命令，则 **_r** 接口将是线程安全的，不为 **_r** 则不是线程安全的。

取消点

如果线程在下列接口中执行，则可能出现取消点：

_pututline()	**accept()**	**aio_suspend()**
bgets()	**bwtmpname()**	**catclose()**
catgets()	**catopen()**	**close()**
closedir()	**closelog()**	**confstr()**
connect()	**copylist()**	**creat()**
ctermid()	**cuserid()**	**dbm_close()**
dbm_delete()	**dbm_fetch()**	**dbm_firstkey()**
dbm_nextkey()	**dbm_open()**	**dbm_store()**
devnm()	**dial()**	**endbwent()**
endgrent()	**endpwent()**	**endusershell()**
endutent()	**endutsent()**	**endutxent()**
fclose()	**fcntl()**	**fdopen()**
fflush()	**fgetc()**	**fgetpos()**

fgetpwent()	**fgets()**	**fgetwc()**
fgetws()	**flock()**	**fopen()**
fprintf()	**fputc()**	**fputs()**
fputwc()	**fputws()**	**fread()**
freopen()	**fscanf()**	**fseek()**
fseeko()	**fsetpos()**	**fstatvfsdev()**
fsync()	**ftell()**	**ftello()**
ftw()	**fwprintf()**	**fwrite()**
fwscanf()	**getbwent()**	**getc()**
getc_unlocked()	**getchar()**	**getchar_unlocked()**
getcwd()	**getdate()**	**getgrent()**
getgrgid()	**getgrgid_r()**	**getgrnam()**
getgrnam_r()	**getlogin()**	**getlogin_r()**
getmsg()	**getpmsg()**	**getpwent()**
getpwnam()	**getpwnam_r()**	**getpwuid()**
getpwuid_r()	**gets()**	**gettxt()**
getusershell()	**getutent()**	**getutid()**
getutline()	**getutxent()**	**getutsent()**
getutsid()	**getutsline()**	**getutspid()**
getutxid()	**getutxline()**	**getw()**
getwc()	**getwchar()**	**getwd()**
glob()	**globfree()**	**iconv()**
iconv_close()	**iconv_open()**	**initgroups()**
insque()	**ioctl()**	**isastream()**
lckpwdf()	**lockf()**	**lockf64()**
lseek()	**lseek64()**	**mkstemp()**
msgrcv()	**msgsnd()**	**msync()**
nanosleep()	**nftw()**	**nftw2()**
open()	**opendir()**	**openlog()**
pause()	**pclose()**	**perror()**
pfmt()	**poll()**	**popen()**
pread()	**printf()**	**pselect()**
pthread_cond_timedwait()	**pthread_cond_wait()**	**pthread_join()**
pthread_testcancel()	**putc()**	**putc_unlocked()**
putchar()	**putchar_unlocked()**	**putmsg()**
putpmsg()	**putpwent()**	**puts()**
pututline()	**pututsline()**	**pututxline()**
putw()	**putwc()**	**putwchar()**

putws()	**pwrite()**	**read()**
readdir()	**readdir_r()**	**readv()**
recv()	**recvfrom()**	**recvmsg()**
remove()	**remque()**	**rename()**
rewind()	**rewinddir()**	**scandir()**
scanf()	**seekdir()**	**select()**
semop()	**send()**	**sendmsg()**
sendto()	**sendfile()**	**setbwent()**
setgrent()	**setpwent()**	**setusershell()**
setutent()	**setutsent()**	**setutxent()**
sigpause()	**sigsuspend()**	**sigtimedwait()**
sigwait()	**sigwaitinfo()**	**sleep()**
socket()	**socketpair()**	**statvfsdev()**
strerror()	**syslog()**	**system()**
t_accept()	**t_alloc()**	**t_bind()**
t_close()	**t_connect()**	**t_error()**
t_getinfo()	**t_getprotaddr()**	**t_getstate()**
t_listen()	**t_look()**	**t_open()**
t_optmgmt()	**t_rcv()**	**t_rcvconnect()**
t_rcvdis()	**t_rcvrel()**	**t_rcvudata()**
t_rcvuderr()	**t_snd()**	**t_snddis()**
t_sndrel()	**t_sndudata()**	**t_strerror()**
t_sync()	**t_unbind()**	**tcdrain()**
tmpfile()	**tmpnam()**	**ttyname()**
ttyname_r()	**ttyslot()**	**ulckpwdf()**
undial()	**ungetc()**	**ungetwc()**
unlink()	**updatebwdb()**	**usleep()**
utmpname()	**vfprintf()**	**vfscanf()**
vfwprintf()	**vpfmt()**	**vprintf()**
vscanf()	**vwprintf()**	**wait()**
wait3()	**waitid()**	**waitpid()**
wordexp()	**wordfree()**	**wprintf()**
write()	**writev()**	**wscanf()**

注释：当某线程在 UNIX 2003 兼容应用程序中的 **isastream()** 内执行时，不会出现取消点。

取消安全

下列所有接口都是取消安全的：

accept()	**bind()**	**connect()**	**dn_comp()**
dn_expand()	**endhostent()**	**endnetent()**	**endprotoent()**
endservent()	**get_resfield()**	**gethostbyaddr()**	**gethostbyname()**
gethostent()	**getmsg()**	**getnetbyaddr()**	**getnetbyname()**
getnetent()	**getpeername()**	**getpmsg()**	**getprotobyname()**
getprotobynumber()	**getprotoent()**	**getservbyname()**	**getservbyport()**
getservent()	**getsockname()**	**getsockopt()**	**herror()**
inet_addr()	**inet_lnaof()**	**inet_makeaddr()**	**inet_netof()**
inet_network()	**inet_ntoa()**	**isastream()**	**listen()**
msync()	**net_aton()**	**net_ntoa()**	**poll()**
putmsg()	**putpmsg()**	**rcmd()**	**recv()**
recvfrom()	**recvmsg()**	**res_init()**	**res_mkquery()**
res_query()	**res_search()**	**res_send()**	**rexec()**
rresvport()	**ruserok()**	**sbrk()**	**send()**
sendfile()	**sendmsg()**	**sendto()**	**set_resfield()**
sethostent()	**setnetent()**	**setprotoent()**	**setservent()**
setsockopt()	**shutdown()**	**socket()**	**socketpair()**

异步信号安全

下列接口所有都是异步信号安全的:

_exit()	**accept()**	**access()**	**alarm()**
bind()	**cfgetispeed()**	**cfgetospeed()**	**cfsetispeed()**
cfsetospeed()	**chdir()**	**chmod()**	**chown()**
close()	**connect()**	**creat()**	**dup()**
dup2()	**execle()**	**execve()**	**fcntl()**
flock()	**fpathconf()**	**fstat()**	**getegid()**
geteuid()	**getgid()**	**getgroups()**	**getmsg()**
getpeername()	**getpgrp()**	**getpid()**	**getpmsg()**
getppid()	**getsockname()**	**getsockopt()**	**getuid()**
isastream()	**kill()**	**link()**	**listen()**
lseek()	**memset()**	**mkdir()**	**mkfifo()**
msync()	**net_aton()**	**net_ntoa()**	**open()**
pathconf()	**pause()**	**pipe()**	**poll()**
putmsg()	**putpmsg()**	**raise()**	**read()**
recv()	**recvfrom()**	**recvmsg()**	**rename()**
rmdir()	**send()**	**sendfile()**	**sendmsg()**
sendto()	**setgid()**	**setpgid()**	**setsid()**

setsockopt()	**setuid()**	**shutdown()**	**sigaction()**
sigaddset()	**sigdelset()**	**sigemptyset()**	**sigfillset()**
sigismember()	**signal()**	**sigpending()**	**sigprocmask()**
sigqueue()	**sigsuspend()**	**sleep()**	**socket()**
socketpair()	**stat()**	**sysconf()**	**tcdrain()**
tcflow()	**tcflush()**	**tcgetattr()**	**tcgetpgrp()**
tcsendbreak()	**tcsetattr()**	**tcsetpgrp()**	**time()**
times()	**umask()**	**uname()**	**unlink()**
utime()	**wait()**	**waitpid()**	**write()**

注释： **isastream()** 在 UNIX 2003 兼容应用程序中不是异步信号安全的。

异步取消安全

下列所有接口都是异步取消安全的：

a64l()	**abs()**	**accept()**
addmntent()	**atof()**	**atoi()**
atol()	**bcmp()**	**bcopy()**
bind()	**bsearch()**	**bzero()**
cfgetispeed()	**cfgetospeed()**	**cfsetispeed()**
cfsetospeed()	**connect()**	**div()**
execv()	**execve()**	**ffs()**
fnmatch()	**fsetaclentry()**	**ftok()**
getclock()	**getmsg()**	**getpeername()**
getpmsg()	**getrlimit()**	**getsockname()**
getsockopt()	**getsubopt()**	**index()**
insque()	**isalnum()**	**isalpha()**
isascii()	**isastream()**	**iscntrl()**
isdigit()	**isgraph()**	**islower()**
isprint()	**ispunct()**	**isspace()**
isupper()	**iswalnum()**	**iswalpha()**
iswcntrl()	**iswctype()**	**iswdigit()**
iswgraph()	**iswlower()**	**iswprint()**
iswpunct()	**iswspace()**	**iswupper()**
iswxdigit()	**isxdigit()**	**l3tol()**
labs()	**ldiv()**	**lfind()**
listen()	**lsearch()**	**ltol3()**
mblen()	**mbstowcs()**	**mbtowc()**
memccpy()	**memchr()**	**memcmp()**

memcpy()	**memmove()**	**memset()**
mkfifo()	**mktemp()**	**msem_init()**
msem_lock()	**msgrcv()**	**msgsnd()**
msync()	**nanosleep()**	**net_aton()**
net_ntoa()	**pathfind()**	**pause()**
poll()	**pstat()**	**pstat_getlwp()**
pthread_cancel()	**pthread_setcancelstate()**	**pthread_setcanceltype()**
putmsg()	**putpmsg()**	**recv()**
recvfrom()	**recvmsg()**	**remque()**
rindex()	**rmdirp()**	**sbrk()**
semop()	**send()**	**sendfile()**
sendmsg()	**sendto()**	**setaclentry()**
setcat()	**setclock()**	**setrlimit()**
setsockopt()	**shutdown()**	**sigsuspend()**
socket()	**socketpair()**	**strcasecmp()**
strcat()	**strchr()**	**strcmp()**
strcoll()	**strcpy()**	**strcspn()**
strdup()	**strlen()**	**strncasecmp()**
strncat()	**strncmp()**	**strncpy()**
strord()	**strpbrk()**	**strrchr()**
strrstr()	**strspn()**	**strstr()**
strtod()	**strtok_r()**	**strtol()**
strtold()	**strtoul()**	**strxfrm()**
swab()	**tcdrain()**	**tcflow()**
tcflush()	**tcgetattr()**	**tcgetpgrp()**
tcgetsid()	**tcsendbreak()**	**tcsetattr()**
tcsetpgrp()	**toascii()**	**tolower()**
toupper()	**towlower()**	**towupper()**
ulimit()	**wait()**	**wait3()**
waitid()	**wcscat()**	**wcschr()**
wcscmp()	**wcscpy()**	**wcscspn()**
wcslen()	**wcsncat()**	**wcsncmp()**
wcsncpy()	**wcspbrk()**	**wcsrchr()**
wcsspn()	**wcsstr()**	**wcstod()**
wcstok()	**wcstol()**	**wcstombs()**
wcstoul()	**wcswcs()**	**wctomb()**
wctype()		

另请参阅

intro(3C)、 pthread(3T)、 signal(5)。

请参考 **http://www.devresource.hp.com** 和 **http://docs.hp.com** 网站上的线程白皮书。

名称

timeslice - 调度间隔（以每秒经过的时钟周期为单位）

值

无故障

(HZ/10)

HZ 定义了系统配置每秒经过的时钟周期的个数。

缺省值

(HZ/10)

其中 **HZ** 等于 **100** 。

允许值

允许使用范围 **-1** 到 **2147483647** 的任意值。

值 **-1** 表明没有基于调度抢占的时间片，所有线程将持续执行直到它们主动退出 ，或者由更高优先级的线程将它们抢占了为止。

建议值

普通情况下选用缺省值。在需要进行快速循环调度的特殊情况下，可以使用值 **1** 。然而，值的更改可能会对系统性能产生直接影响。客户在改变产品系统的值之前，必须评估其工作负荷环境下对系统性能的影响。

说明

timeslice 可调参数定义了调度时间间隔，即内核时间调度程序从线程中切换到其他环境以运行其他相同优先级的线程之前，该线程在处理器上的执行时间。当线程在处理器上开始执行时，该线程将设置为运行 **timeslice** 可调参数中的时钟周期数。在每一个发现执行中线程的时钟中断处, 对应于该线程的时间平衡量将递减,当此平衡量为零时,此线程将切换到其他环境。

timeslice 值控制操作系统实现的用户抢占的一个方法。较大的值将减少运行线程的抢占；但是有许多其他的原因引起线程的用户抢占，并且 **timeslice** 可调参数将无法在此进行控制。

对 **timeslice** 值的更改可能会对系统吞吐量及响应时间产生直接影响。过小的值可能会导致过多的环境切换，而过大的值则可能会导致可运行线程不足。

应该由谁来更改此可调参数?

任何用户。

对于更改的限制

对此可调参数的更改将在下次重新引导时生效。

何时应增加此可调参数的值?

由于 **timeslice** 可调参数可全局应用于系统中的所有线程（ **sched_fifo** 除外），并且与它们的调度策略和优先级无关。因此对此可调参数值的任意增加都会对所有线程产生相同的时间量的增加。

如果系统中由于抢占（由较高优先级的线程产生）而导致过多的环境切换，则可以增加该可调参数的值以使低优

先级的线程在获得时间调度时得到更多的执行时间，因为当它们可运行时，较高优先级的线程将抢占较低优先级线程的运行时间。

增加此值的副作用是什么？

增加 **timeslice** 可调参数的值可能会导致一些线程不足，因为它们必须等待更久以来得执行的机会。这可能会导致性能处理问题。

何时应降低此可调参数的值？

如果在增加额外环境切换的情况下，要求有更快速的周转响应时间，则 **timeslice** 可调参数的值应较低。当系统没有太多的计算机密集型应用程序时，线程将阻塞并且更频繁地进行抢占，而没有利用它们的完整时间量。

降低此值的副作用是什么？

降低 **timeslice** 可调参数值将会导致更多的环境切换，从而增加花费在系统空间中的时间，同时减少用于用户空间的时间。同时，计算密集型的应用程序将受到性能降低产生的影响。

同时还应更改哪些其他可调参数的值？

无。

警告

所有 HP-UX 内核可调参数都特定于各个版本。将来的 HP-UX 版本中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅网站 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

timeslice 由 HP 开发。

名称

timezone, dst - 国际标准时间（格林威治时间）和本地时间之差

值

缺省值

timezone = **420**

dst = **1**

缺省值 **1** 表示时区可调参数的缺省值（英格兰格林威治以西 420 分钟）应当解释为夏令时值。

允许值

-720 到 **720** 。

可调参数 **timezone** 的值应该对应于一个已定义的时区，因此应该至少是 15 分钟的整数倍。更多的情况下此值会为 60 或 30 分钟的整数倍，对应一小时或半小时时区。

可调参数 **dst** 的值指定是否采用夏令时。如果值设置为 **1** ，则表示系统使用夏令时，而如果值设置为 **0** ，则表示系统不使用夏令时。

建议值

同样建议使用任何允许值。但是，所选择的取值应该对应系统站点的时区和夏令时方案，或者对应对于应用程序或用户有实际意义的时区和夏令时方案。

说明

可调参数 **timezone** 是格林威治标准时间（国际标准时间）与本地的时间的差，表示为英格兰格林威治以西的分钟数。可调参数 **dst** 表示将可调参数 **timezone** 解释为标准时间还是夏令时时间的值。

这些可调参数提供了一种方法，用于在格林威治标准时间（国际标准时间）和本地时间之间进行转换。虽然 **timezone** 是一个可调参数，但是它不能用于影响系统行为。其实，它是在重新引导系统时，用于记录时区信息的。这一信息可以通过 HP-UX 扩展的 **gettimeofday** 操作返回。可调参数 **timezone** 独立于其他表示时区的方法。例如， **date** 通过使用环境变量 **TZ** 来获取或设置国际标准时间或本地时间。在这种情况下，可调参数 **timezone** 不起作用。通常情况下，关于特定系统调用，最好参考联机帮助页（例如 **localtime**() 和 **tzset**() ），检查是否和可调参数 **timezone** 相关。

应该由谁来更改此可调参数?

任何用户。

对于更改的限制

对此可调参数的更改将在下次重新引导时生效。

何时应更改此可调参数的值?

这些可调参数可以更改以匹配系统的地理时区或者用户时区。

同时还应更改哪些其他可调参数的值?

可调参数 **timezone** 和 **dst** 应该总是设置和解释为一对值。

警告

所有 HP-UX 内核可调参数都是针对于特定版本的。将来的 HP-UX 版本中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装后，某些可调参数可能不再是缺省值或建议值。有关安装对可调参数值影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅网站 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

timezone 和 **dst** 由 HP 开发。

名称

types - 原始系统数据类型

概要

#include <sys/types.h>

说明

备注

本页中给出的示例是一个典型版本。通常情况下都应该有类型名称，虽然在例外情况下（如果有的话）可以在相关内容中进行描述。大多数情况下，实现类型定义的基本类型与实现相关，只要无须更改使用这些类型定义的源代码。在某些情况下，类型定义实际上是对常用类型的快捷方式，而且不会发生变化。

包含文件中所定义的数据类型用于 HP-UX 系统代码；用户代码可访问一部分这些类型的数据：

```
typedef struct { int r[1]; }  *physadr;
typedef char                 *caddr_t;
typedef unsigned int          uint;
typedef unsigned short        ushort;
typedef unsigned long         ino_t;
typedef short                 cnt_t;
typedef long                  time_t;
typedef long                  dev_t;
typedef long                  off_t;
typedef long                  paddr_t;
typedef long                  key_t;
typedef int32_t               pid_t;
typedef long                  uid_t;
typedef long                  gid_t;
typedef long                  blkcnt_t;
```

请注意，上面定义的名称是已经过标准化的，但是所定义的实际类型会随着的 HP-UX 实现的不同而有所变化。

类型的含义包括：

physadr 用作指向内存的指针；这一指针将调整以遵循与硬件相关的指令寻址约定。

caddr_t 作为无类型指针或者指向无类型内存的指针。

uint 无符号整数的快捷方式。

ushort 无符号短整数的快捷方式。

ino_t 用于指定 I 编号。 HP-UX 11i 包含的所有本地文件系统（包括 HFS 和 VxFS 3.5），都使用适用于 32 位系统的值。一些远程 NFS 服务器可以使用较大的值，这些值将被截断，而不会对 32 位应用程序造成错误，并且可能不会生成唯一值。

cnt_t 用于一些实现中，以保留某些内核数据结构的引用计数。

time_t 自 1970 年 1 月 1 日国际标准时间 00:00:00 起编码的时间，以秒为单位。

dev_t 指定设备的类型和单元数，由主次两部分编码组成。

off_t 从文件开头计量的偏移量（以字节为单位）。如果使用 **-D_FILE_OFFSET_BITS=64** 或 **-D_LARGEFILE64_SOURCE** 编译 32 位应用程序，则 *off_t* 将成为 *int64_t* 。

paddr_t 用作整型，其大小准确设定为容纳一个指针。

key_t 用于获取消息队列、信号量、或者共享内存标识符的关键字类型，请参阅 *stdipc*(3C) 。

pid_t 用于指定进程和进程组标识符。

uid_t 用于指定用户标识符。

gid_t 用于指定组标识符。

blkcnt_t 以块为单位计量的磁盘限额或者传输大小。如果使用 **-D_FILE_OFFSET_BITS=64** 或 **-D_LARGEFILE64_SOURCE** 编译 32 位应用程序，则 *blkcnt_t* 将成为 *int64_t* 。

符合的标准

<sys/types.h>：AES、SVID3、XPG2、XPG3、XPG4、FIPS 151-2、POSIX.1

名称

uname_eoverflow - 如果该字段的值不恰当，则 **uname()** 系统函数返回 [EOVERFLOW]

值

保证安全

1 - 启用

缺省值

1 - 启用

允许值

0 - 禁用

1 - 启用

说明

uname_eoverflow 可调参数用于控制在程序调用非扩展版本的 **uname()** 时是否对相应字段中出现不恰当的值这一现象发出警报。如果该可调参数的值为 0，那么，在任何返回值被截断的情况下， **uname()** 非扩展版本均不作提示。该可调参数的值为 1 时，如果发生截断，则将返回非零值，并将 **errno** 设置为 [EOVERFLOW] ，以便向调用方发出警报。

更改此可调参数不会影响 **uname()** 返回的当前系统值。

在将来的 HP-UX 发行版中，当报告了 [EOVERFLOW] （或其他错误）时， **uname()** 函数将不返回任何数据。有关 **uname()** 的信息，请参阅 *uname*(2) 联机帮助页。

更改此可调参数的人员

如果必须根据错误更改程序行为，则系统管理员可以更改此可调参数。

更改限制

此可调参数是动态的。对于 **uname()** 非扩展版本的所有后续调用，更改均有效。

应该何时增加此可调参数的值

如果需要就 **uname()** 返回的错误向应用程序发出警报，则可以将此可调参数增加到 1。

增加此值的负面影响

如果任何值不适合其非扩展结构的相应字段，则将可调参数值增加到 1 会导致 **uname()** 系统函数的非扩展版本返回非零状态和 [EOVERFLOW] 错误。

应该何时降低此可调参数的值

如果必须允许应用程序处理截断的值，应该将此可调参数降低至 0。当应用程序因错误而异常中止，但仍能接受截断值时，将出现此情况；例如，值对于其操作而言并不重要。

降低此值的负面影响

将可调参数值降低到 0 会导致 **uname()** 系统函数的非扩展版本以无提示方式截断任何不适合非扩展字段的值。发生这种截断时，该函数不会向调用方提示错误警报。

应该同时更改的其他可调参数值

无。

警告

将此可调参数设置为 0（禁用）意味着不会因非扩展 **uname()** 系统调用返回的截断值而向应用程序发出警报。使用这种截断值的应用程序可能会以不确定的方式失败。

在 HP-UX 11i v3 初始版本之前，可能仅会截断 **uname()** 返回的节点名称值，因为该函数返回的其他字段均不接受超过非扩展字段大小的值。但是，以后的更新和发行版可能会扩展其他值，并且也可能会截断其相应的非扩展字段（或导致报告 [EOVERFLOW] 错误）。这种扩展可由系统管理员选择控制。但是，对程序进行升级以使用 **uname()** 系统函数的扩展版本则可以避免将来出现的问题。

有关用法的详细信息，请参阅标题为《Node and Host Name Sizes on HP-UX: Using the Expanded Capabilities》的白皮书。

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

uname_eoverflow 由 HP 开发。

另请参阅

uname(2)、 expanded_node_host_names(5)。

《Node and Host Name Sizes on HP-UX: Using the Expanded Capabilities》白皮书，可从 **http://docs.hp.com** 上获得。

名称

unctrl - unctrl 的定义

说明

头文件 **<unctrl.h>** 定义了 **chtype** 类型，如 **<curses.h>** 中所定义。

下列内容声明为函数，也可定义为宏：

```
char *unctrl(chtype c);
```

另请参阅

<curses.h>。

历史变更记录

在 X/Open Curses 第 4 期中首次发布。

名称

unistd：unistd.h - 标准的结构和符号常量

概要

#include <unistd.h>

说明

头文件 **<unistd.h>** 定义了如下结构和符号常量:

access() 函数的符号常量为:

R_OK	测试读取权限。
W_OK	测试写入权限。
X_OK	测试执行（搜索）权限
F_OK	测试文件是否存在。

所有常量 **F_OK**、**R_OK**、**W_OK**、**X_OK**，以及表达式 **R_OK|W_OK**、**R_OK|X_OK** 和 **R_OK|W_OK|X_OK** 都有不同的值。

表示一个空指针的符号常量:

NULL

lseek() 和 **fcntl()** 函数的符号常量（下列常量有不同的值）：

SEEK_SET	设置文件地址偏移量为“offset”。
SEEK_CUR	设置文件地址偏移量为当前值加上“offset”。
SEEK_END	设置文件地址偏移量为 EOF 加上“offset”。

符号常量（有固定值）：

_POSIX_VERSION	表示实现 *IEEE Std 1003.1* 版本标准的整数值。当前值是 199009L，表示年（4 位）和月（2 位），此标准已被 IEEE Standards Board 认可。然而，如果符号 **_AES_SOURCE**、**_XPG3** 或 **_POSIX1_1988** 中的任何一个在包括 **<unistd.h>** 之前已定义，则这个符号的值将是 198808L。
_POSIX2_VERSION	表示 *IEEE Std*1003.2 版本标准实现的整数值。当前值是 199209L，表示年（4 位）和月（2 位），此标准已被 IEEE Standards Board 认可。
_POSIX2_C_VERSION	表示实现 *IEEE Std 1003.2* 版本 C 语言绑定选项的整数值。当前值是 199209L，表示年（4 位）和月（2 位），此标准已被 IEEE Standards Board 认可。
_XOPEN_VERSION	表示实现 X/Open 可移植性指南的期号的整数值。当前值是 4，表示第 4 期。然而，如果符号 **_XPG3** 在包括 **<unistd.h>** 之前定义，则此符号的值将为 3。

如果编译后对应选项或限制的状态没有发生变化，下列符号常量会在头文件中定义。如果一个符号不在此头文件

中，则相应选项或限制的值或出现应当在 **sysconf()** 或 **pathconf()** 的执行时间中确定：

_POSIX_CHOWN_RESTRICTED
chown() 的使用仅限于有相应权限的进程。

_POSIX_JOB_CONTROL 实现支持作业控制（所有 HP-UX 实现为 true）。

_POSIX_NO_TRUNC 比 **NAME_MAX** 更长的路径名组成部分将生成错误。

_POSIX_REALTIME_SIGNALS
实现支持实时信号扩展（所有 HP-UX 实现为 true）。

_POSIX_SAVED_IDS 跨 **exec()** 调用保存有效的用户和组（所有 HP-UX 实现为 true）。

_POSIX_FSYNC 实现支持文件同步（所有 HP-UX 实现为 true）。请参阅 *open*(2)。

_POSIX_SYNCHRONIZED_IO
实现支持同步 IO（所有 HP-UX 实现为 true）。请参阅 *open*(2)。

_POSIX_VDISABLE 可以使用此字符禁用终端特殊字符（请参阅 *termio*(7)）。

_POSIX_THREADS 实现支持 POSIX 线程。

_POSIX2_C_BIND 所有 POSIX.2 C 语言功能在 **c89** C 编译程序（请参阅 *cc*(1)）使用的缺省库中提供。

_POSIX2_LOCALEDEF 可以使用 **localedef** 命令（请参阅 *localedef*(1M)）定义新的语言环境。

_POSIX2_UPE 系统支持 *IEEE Std*1003.2a （POSIX 用户可移植性实用工具选项）。

_POSIX2_CHAR_TERM 至少存在一个终端支持所有需要的 POSIX.2a 命令。

所有名称以 **_CS**、 **_PC** 和 **_SC** （请参阅 *confstr*(3C)、 *pathconf*(2) 和 *sysconf*(2)）开始的符号常量将被定义。

定义了下列文件流的符号常量：

STDIN_FILENO 标准输入 (*stdin*) 的文件编号。
STDOUT_FILENO 标准输出 (*stdout*) 的文件编号。
STDERR_FILENO 标准错误 (*stderr*) 的文件编号。

将定义类型 **size_t**、 **ssize_t**、 **uid_t**、 **gid_t**、 **off_t** 和 **pid_t**。

将为下列函数提供声明：

access()	**alarm()**	**brk()**	**chdir()**
chown()	**chroot()**	**close()**	**confstr()**
crypt()	**ctermid()**	**cuserid()**	**dup()**
dup2()	**encrypt()**	**endusershell()**	**execl()**
execle()	**execlp()**	**execv()**	**execve()**

execvp()	**_exit()**	**fchown()**	**fork()**
fpathconf()	**fsync()**	**ftruncate()**	**getcwd()**
getegid()	**geteuid()**	**getgid()**	**getgroups()**
gethostname()	**getlogin()**	**getopt()**	
getpass()	**getpgrp()**	**getpgrp2()**	**getpid()**
getppid()	**getuid()**	**getusershell()**	**initgroups()**
ioctl()	**isatty()**	**link()**	**lockf()**
logname()	**lseek()**	**mkstemp()**	**mktemp()**
nice()	**pathconf()**	**pause()**	**pipe()**
prealloc()	**read()**	**readlink()**	**rmdir()**
sbrk()	**setgid()**	**setgroups()**	**sethostname()**
setpgid()	**setpgrp()**	**setpgrp2()**	**setresgid()**
setresuid()	**setsid()**	**setuid()**	**setusershell()**
sgetl()	**sleep()**	**sputl()**	**swab()**
swapon()	**symlink()**	**sync()**	**sysconf()**
tcgetpgrp()	**tcsetpgrp()**	**truncate()**	**ttyname()**
ttyslot()	**unlink()**	**vfork()**	**write()**

作者

unistd 由 HP 开发。

另请参阅

access(2)、 chown(2)、 exit(2)、 fcntl(2)、 kill(2)、 lseek(2)、 open(2)、 pathconf(2)、 sysconf(2)、 confstr(3C)、 limits(5)、 stdsyms(5)、 termio(7)。

符合的标准

<unistd.h>：AES、SVID3、XPG2、XPG3、XPG4、FIPS 151-2、POSIX.1、POSIX.2、POSIX.4

已过时

名称

unlockable_mem - 已过时的内核可调参数

说明

unlockable_mem 可调参数已过时并被删除。

内存锁定允许具有权限的用户指定哪些页需要保留内存中，而且不受交换进程的影响。利用这一功能，可以确保内存分页和交换产生的延迟不会影响内存访问时间。例如，锁定是为物理内存缺乏的系统上的具有权限的用户提供的一个工具。具有权限的进程不像其余进程一样交换，相反，它们可以锁定它们的部分地址空间。一旦具有权限的进程的内存页被锁定后，它们就无需再担心内存竞争。但是，没有权限的进程不得不争用内存。

unlockable_mem 通过设置不能被用户进程锁定的内存量，对这种利用权限的行为提供限制。

警告

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

unlockable_mem 由 HP 开发。

名称

unwind: unwind.h - 堆栈辗转开解库入口点和便捷宏的概述

概要

#include <unwind.h>

说明

<unwind.h> 标头定义了堆栈辗转开解库的应用程序编程接口 (API)，即基于 Itanium 的系统上的针对 HP-UX 的 **libunwind.so** 。本联机帮助页讨论了堆栈辗转开解的通用概念和应该如何使用堆栈辗转开解库。这旨在补充用于描述辗转开解库入口点的第 3X 节联机帮助页。本联机帮助页也解释了辗转开解标题的格式和包含在基于 Itanium 的可执行文件（此文件不隐藏在《Itanium Processor Family Software Conventions and Runtime Architecture》和相关文档中）的辗转开解表中的一些详细信息。

术语的概念与定义

区分任何高级程序语言与汇编语言的最基本特征之一就是过程或函数调用的内置支持。已被调用过的过程自身可调用其他过程，形成一个过程 “调用链” 。术语 “调用方” 和 “被调用方” 描述了过程调用链中过程间的调用关系。调用方是调用被调用方的过程。调用链中的一个过程完成执行语句时，它的调用方将恢复执行该调用后的下个语句。

多任务操作系统的基本特征之一是中断事件，受控制的中断流的应用，保存了全部用户可见的处理器状态，并对 “中断处理” 函数进行控制。用两个术语 “中断过程” 和 “处理程序” 来描述过程调用链中过程间的中断关系。

有时用术语 “后继对象” 和 “前置对象” 概括过程调用和中断关系。在过程调用关系中， “前置对象” 指的是调用方，而 “后继对象” 指的是被调用方。在中断关系中， “前置对象” 指的是中断过程，而 “后继对象” 指的是处理程序。

概念上，对于每一个在嵌套调用或中断中处于活动状态的过程，有一个 “激活记录” ，它包含在调用或中断事件时此过程的用户可见的处理器状态。在过程完成后，在调用和删除此过程时，激活记录像被置入一个链中的链接一样。激活记录将以后进先出顺序添加和删除，因此调用链可在技术上描述为 “堆栈” 。

堆栈辗转开解可描述充分恢复前置对象（调用方或中断过程）的激活记录的任务，此前置对象给定其后继对象（被调用方或处理程序）的激活记录，使用户可以使用前置对象的激活记录中的值，将处理器状态初始化为前置对象的状态，随后处理器可继续执行前置对象的指令，就像对被调用方的调用已经返回或中断事件已经完成。

软件应用程序编写者执行堆栈辗转开解任务的动机，可能由于很多原因，包括：

- 实现用户空间调试器的堆栈回溯跟踪命令，该命令显示程序调用堆栈，例如 GNU 调试器中的 **bt** 命令。请参阅 *gdb*(1) 。
- 出于处理例外、乱码收集，或可能为了性能、基于配置文件的工具或其他资源分析的目的，对程序调用堆栈进行检查。
- 显示从错误处理例行程序到 **stderr** 的堆栈追踪码，以简化服务或调试应用程序。

给定过程的激活记录的堆栈辗转开解库形式包含在过程的 **_Unwind_Context** 数据结构中。在过程调用（内存）堆栈中分配给过程的空间称为过程的 “堆栈帧” 。堆栈帧存储了许多信息，这些信息包含在过程的以前定义的概念

性激活记录中。由于此条件为真，通常术语 “帧”被用于代替 “激活记录”，这贯穿于堆栈辗转开解库文档。

辗转开解联机帮助页假定熟悉《Itanium Processor Family Software Conventions and Runtime Architecture》和定义在 5.1 节的《Register Usage》和 11 章中的《Stack Unwinding and Exception Handling》中的术语。

功能

堆栈辗转开解库提供一个 API，用于辗转开解遵循《Processor Family Software Conventions and Runtime Architecture》的程序的堆栈。进程可使用堆栈辗转开解库

- 将它自己的堆栈追踪码输出到标准错误或输出流，这通常用于诊断目的。请参阅 *U_STACK_TRACE*(3X) 和 *_UNW_STACK_TRACE*(3X) 。

- 辗转开解它自己的堆栈、目标进程的堆栈或核心文件的堆栈。对于过程调用堆栈中的大多数激活记录，允许查询保留寄存器值的 “辗转开解环境”。对某些激活记录，也可以查询临时寄存器值。

  要辗转开解一个不是自身或核心文件的进程，堆栈辗转开解库将依赖于客户端来获取辗转开解段位置，并提供一个读取函数来访问其他进程的或核心文件的内存。其实现方法是使用回调函数，客户端必须用堆栈辗转开解库注册此函数。

- 提供堆栈辗转开解支持，以遵循 ANSI C++ 异常处理，包括将控制传递给封闭（嵌套不深）的过程。请参阅《C++ ABI for IA-64》，(Draft)，2000 年 11 月 7 日，位于 **http://www.codesourcery.com/cxx-abi/abi.html**。

为了版本间的兼容性，堆栈辗转开解库接口完全由编程实现，以隐藏数据结构实现的详细信息。

类型定义

API 函数的返回值是一个枚举 **_UNW_ReturnCode** ，具有如下含义：

_UNW_STEP_KERNEL_SAVE_STATE

_Unwind_Context 描述了一个帧，超过该帧后则堆栈辗转开解库不能再步进，这是因为该帧属于内核中断帧，而不是与 **_user_sendsig()** 关联的帧。用户空间辗转开解应用程序应从不接收该返回代码。

_UNW_STEP_BOTTOM

_Unwind_Context 描述一个帧，超过该帧后堆栈辗转开解库不能再步进。当对描述过程的任何 **_Unwind_Context** 调用 **_UNW_step** 时将生成该返回代码，此处描述的过程的帧标有堆栈约定的底线 — 一个保存的 0 返回链接（请参阅《Itanium Processor Family Software Conventions and Runtime Architecture》，第 11.1 章：“辗转开解堆栈”）。

_UNW_OK

一切都正确。

_UNW_STEP_ERROR

步骤中发生的一些常规性问题。

_UNW_STEP_INVALID_IP
_Unwind_Context 中的指令指针值标记为无效。

_UNW_STEP_INVALID_SP
_Unwind_Context 中的堆栈指针值标记为无效。

_UNW_STEP_INVALID_GR
步骤过程中，在 **_Unwind_Context** 中遇到标记为无效的通用寄存器值。

_UNW_STEP_INVALID_PFS
_Unwind_Context 中的 AR.PFS 值标记为无效。

_UNW_STEP_INVALID_RSC
_Unwind_Context 中的 AR.RSC 值标记为无效。

_UNW_STEP_INVALID_BSP
_Unwind_Context 中的 AR.BSP 值标记为无效。

_UNW_STEP_INVALID_BSPSTORE
_Unwind_Context 中的 AR.BSPSTORE 值标记为无效。

_UNW_STEP_INVALID_CFM
_Unwind_Context 中的 AR.BFM 值标记为无效。

_UNW_STEP_INVALID_BR
步骤过程中，在 **_Unwind_Context** 中遇到标记为无效的分支寄存器值。

_UNW_STEP_BAD_BSP_ALIGNMENT
_UNW_currentContext 的值 **AR.BSP** 不是对齐的。

_UNW_STEP_INVALID_RNAT
_Unwind_Context 中的 AR.RNAT 值标记为无效。

_UNW_STEP_NO_DESCRIPTOR_FOR_NON_LEAF
堆栈辗转开解库不能为过程找到辗转开解描述符，堆栈辗转开解库可以证明此过程不是叶过程。

_UNW_STEP_CORRUPT_DESCRIPTOR
帧的辗转开解描述符的格式错误。

_UNW_STEP_RSE_NOT_FLUSHED
将不刷新寄存器堆栈引擎。

_UNW_STEP_SIGNAL_CONTEXT
查询中断保存用户环境时返回错误值。

_UNW_STEP_NOT_ALLOWED_IN_STATE

可以从以下状态之一中调用 **_UNW_step**： *Start* 、 *Bad* 或 *Kernel_Bottom_Frame* 。请参阅下面的“状态”小节。

_UNW_INITIALIZATION_RANGE_ERROR

在允许的一组寄存器的外部初始化的客户端。

_UNW_QUERY_RANGE_ERROR

客户端在允许的一组寄存器的外部查询。

_UNW_QUERY_INVALID_ERROR

客户端查询 **_Unwind_Context** 中标记为无效的寄存器。

_UNW_SET_NOT_ALLOWED_IN_STATE

客户端试图初始化值，但 **_Unwind_Context** 不处于 *Init* 状态。

_UNW_CURRENT_CONTEXT_FAILED

发生在调用 **_UNW_currentContext** 中的常规问题。

_UNW_CURRENT_CONTEXT_NOT_ALLOWED_IN_STATE

客户端从下列状态之一调用 **_UNW_currentContext**： *Start* 、 *Bad* 、 *Stop* 、 *Frame* 。

_UNW_MEMORY_ALLOCATION_ERROR

堆栈辗转开解库不能分配足够的内存以执行请求的函数。

_UNW_CLEAR_NOT_ALLOWED_IN_STATE

客户端从下列状态之一调用 **_UNW_clear**： *Start* 、 *Bad* 、 *Stop* 。

_UNW_QUERY_NOT_ALLOWED_IN_STATE

客户端调用 **_UNW_getKernelSavedContext** ，但不处于 *Kernel_Bottom_Frame* 状态。

_UNW_INTERNAL_ERROR

发生了一些逻辑问题。请联系 HP 支持。

_UNW_STEP_INTERRUPTION_ABI_MISMATCH

堆栈辗转开解库遇到了一个错误的 **abi** 字段，该字段的格式为 **P10** 辗转开解描述符。请参阅《Itanium Processor Family Software Conventions and Runtime Architecture》，第 B.3 节：序言部分的描述符记录。

TRUE 和 FALSE 的表示形式是通过 **_UNW_Boolean** 枚举通信，该枚举具有枚举成员 **_UNW_FALSE** 和 **_UNW_TRUE** 。 **_UNW_FALSE** 为 0 值。

整数值通过如下定义的类型通信:

uint32_t 是无符号 32 位整数。

uint64_t 是有符号 64 位整数。

int32_t 是有符号 32 位整数。

int64_t 是无符号 64 位整数。

值查询和初始化函数使用多个结构传递值。它们如下所述：

_UNW_KernelSavedContext 结构有两个 **uint32_t** 类型的字段：**p10_abi_value** 和 **p10_context_value** 。

_UNW_FR_Value 结构有两个 **uint64_t** 类型的字段：**first_container** 和 **second_container** 。

_UNW_GR_Value 结构有一个类型为 **uint64_t** 、 **value** 的字段，和一个类型为 **_UNW_Boolean**、 **NaT** 的字段。

当查询和设置应用程序寄存器的值时，枚举 **_UNW_AppReg** 由函数 **_UNW_getAR** 和 **_UNW_setAR** 使用。**_UNW_AppReg** 有枚举成员 **_UNW_AR_RSC** 、 **_UNW_AR_BSP** 、 **_UNW_AR_BSPSTORE** 、**_UNW_AR_RNAT** 、 **_UNW_AR_CCV** 、 **_UNW_AR_UNAT** 、 **_UNW_AR_FPSR** 、 **_UNW_AR_ITC** 、**_UNW_AR_PFS** 、 **_UNW_AR_LC** 、 **_UNW_AR_EC** 和 **_UNW_AR_LIST_COUNT** 。

_UNW_AR_FIRST_REG 是 **_UNW_AR_RSC** 的同义名称。

通信与定位辗转开解信息

《Itanium Processor Family Software Conventions and Runtime Architecture》的第 11 章中有基于 Itanium 的辗转开解体系结构的定义。此文档讨论辗转开解机制的需要和必须满足的运行时体系结构约定，并提供辗转开解信息块布局和语义的详细信息。它保留环境相关的详细信息，包括将由操作系统环境定义和处理的定位辗转开解表的方法。堆栈辗转开解库处理客户端的这些环境相关的详细信息。有兴趣学习辗转开解信息的环境相关详细信息的用户可以参考《IA-64 Runtime Architecture Supplement》系列中的文档，其中包括授权的文档《Processor-Specific ELF Supplement for IA-64》。

回调

为了执行堆栈辗转开解的任务，对于任何给予的加载模块（由加载模块中的任何有效指令地址标识），堆栈辗转开解库需要确定文本基准、辗转开解标题位置和全局数据指针的值（通用寄存器 **GP** ）。辗转开解函数（例如 **U_STACK_TRACE()** ）和堆栈辗转开解库的异常处理支持使用 *dlmodinfo*(3C) 获取这些值。使用堆栈辗转开解库执行交叉进程的辗转开解的客户端（例如调试器），需要在堆栈辗转开解库的数据结构构成中注册一个函数来执行在 **dlmodinfo** 处的查找任务。函数有 **_UNW_LoadMapFromIP** 类型。查找函数的语义和接口在 *_UNW_createContext*(3X) 中定义

此外，堆栈辗转开解库需要从目标进程内存（或内存映像，如果进程为死进程）中读取值的功能。用堆栈辗转开解库执行交叉进程辗转开解的客户端需要在数据结构构成中注册一个函数来执行内存读取任务（取代直接内存引用）。函数有 **_UNW_ReadTargetMem** 类型。内存读取函数的语义和接口在 *_UNW_createContext*(3X) 中定义。

客户端/库交互

客户端执行堆栈辗转开解操作的步骤包括

- **_Unwind_Context** 数据结构的构成

- **_Unwind_Context** 的初始化
- 步进 **_Unwind_Context**
- 查询 **_Unwind_Context**
- 可能清除并重新初始化 **_Unwind_Context**
- 最后是破坏 **_Unwind_Context** 。

注释: 例外的客户端使用由 aCC 标准委员会管理的与语言无关的 API 。请参阅 CR http://www.codesourcery.com/cxx-abi/abi.html。

状态

状态机器用于帮助定义堆栈辗转开解操作任务的合法顺序，并描述堆栈辗转开解库对来自客户端的调用的响应。状态机器由八种状态及其状态间的转换定义，这些状态由堆栈辗转开解库 API 函数的调用触发，或在某些情况下由系统条件触发，例如内存条件的输出。状态和可能的转换有:

Start
: 在 **_Unwind_Context** 对象的构成之前。

  从 *Start* ，客户端可调用 **_UNW_createContextForSelf** 和 **_UNW_createContext** 。它们将 **_Unwind_Context** 转换到状态 *Init* 或状态 *Bad* 。

Bad
: 指示 **_Unwind_Context** 不可用的状态。在构成中，错误的尝试后可以进入这种状态，— 最有可能是由内存分配的错误造成。如果在辗转开解中辗转开解库丢失，则也可以进入这种状态，— 例如，遇到错误初始化环境中的错误指针值后。

  除了 **_UNW_destroyContext** ，在 *Bad* 状态中，堆栈辗转开解库的行为未定义。在状态 *Bad* 中，客户端的最安全资源将通过调用 **_UNW_destroyContext** 来破坏 **_Unwind_Context** 对象，它会将 **_Unwind_Context** 转换到状态 *Stop* 。

Init
: 允许客户端用相关用户可见处理器状态（描述程序执行的给定快照）初始化 **_Unwind_Context** 的状态。

  在 *Init* 中，客户端可以调用 **_UNW_setGR** 、 **_UNW_setGR_NaT** 、 **_UNW_setFR** 、 **_UNW_setBR** 、 **_UNW_setIP** 、 **_UNW_setAR** 、 **_UNW_setPR** 、 **_UNW_setPreds** 、 **_UNW_setCFM** 、 **_UNW_jmpbufContext** 和 **_UNW_clear** ，这其中没有一个会导致状态转换。客户端也可以调用 **_UNW_step** 或 **_UNW_currentContext** ，将 **_Unwind_Context** 转换到状态 *Frame* 或 *Bad* 。客户端可以调用 **_UNW_destroyContext** ，将 **_Unwind_Context** 转换到状态 *Stop* 。

Frame
: 通过 **_UNW_step** 或 **_UNW_currentContext** ，堆栈辗转开解库在其中生成或处理处理器状态的 **_Unwind_Context** 描述的状态。客户端可以查询并获得大多数保留的寄存器值的有效值。在 *Frame* 状态中，大多数临时寄存器值是无效的。请注意，任何状态都允许使用查询函数，但是仅保证在下面的“查询”一节列出的某些状态中返回值的有效性。

在 *Frame* 中，客户端可以调用 **_UNW_currentContext** ，将 **_Unwind_Context** 转换到状态 *Frame* 或 *Bad* 。它可以调用 **_UNW_step** ，将 **_Unwind_Context** 转换到状态 *Frame* 、 *Bad* 、 *Kernel_Bottom_Frame* 或 *User_Sendsig_Frame* 。客户端可以调用 **_UNW_clear** ，将 **_Unwind_Context** 转换到状态 *Init* 。客户端可以调用 **_UNW_destroyContext** ，将 **_Unwind_Context** 转换到状态 *Stop* 。

User_Sendsig_Frame

_Unwind_Context 在其中描述包装函数（内核通过该函数调用用户的信号处理程序）的处理器状态的状态。请参阅《Itanium Processor Family Runtime Architecture Supplement》、《Signal Handling on HP-UX》。从 *User_Sendsig_Frame* ，客户端可以调用 **_UNW_currentContext**，将 **_Unwind_Context** 转换到状态 *Frame* 或 *Bad* 。它可以调用 **_UNW_step** ，将 **_Unwind_Context** 转换到状态 *User_Interrupted_Frame* 或 *Bad* 。客户端可以调用 **clear** ，将 **_Unwind_Context** 转换到状态 *Init* 。客户端可以调用 **_UNW_destroyContext** ，将 **_Unwind_Context** 转换到状态 *Stop* 。

User_Interrupted_Frame

_Unwind_Context 在其中描述被信号中断的用户代码的状态。这种状态的唯一特征是客户端可以查询并获得临时分支、临时判定、临时浮点和临时通用寄存器以及保留寄存器的有效值。在一系列对 **_UNW_step** 的调用中，与状态 *User_Interrupted_Frame* 关联的帧总是位于与状态 *User_Sendsig_Frame* 关联的帧后。

从 *User_Interrupted_Frame* ，客户端可以调用 **_UNW_currentContext** 或 **_UNW_step** ，将 **_Unwind_Context** 转换到状态 *Frame* 或 *Bad* 。客户端可以调用 **clear** ，将 **_Unwind_Context** 转换到状态 *Init* 。客户端可以调用 **_UNW_destroyContext** ，将 **_Unwind_Context** 转换到状态 *Stop* 。

Kernel_Bottom_Frame

指示内核调用堆栈的底线的状态。在这种状态时，客户端可以使用 **_UNW_getKernelSavedContext** 。请参阅 *_UNW_getKernelSavedContext*(3X) 。

从 *Kernel_Bottom_Frame* ，客户端可以调用 **_UNW_currentContext** 或 **_UNW_step** ，将 **_Unwind_Context** 转换到状态 *Frame* 或 *Bad* 。客户端可以调用 **clear** ，将 **_Unwind_Context** 转换到状态 *Init* 。客户端可以调用 **_UNW_destroyContext** ，将 **_Unwind_Context** 转换到状态 *Stop* 。

Stop **_Unwind_Context** 被忽略。

构成

通过调用 **_UNW_createContextForSelf** 或 **_UNW_createContext** 可以执行构成， **_UNW_createContextForSelf** 主要在进程辗转开解它自己的堆栈时使用。 **_UNW_createContext** 主要在进程辗转开解不同进程的堆栈或保留在核心文件中的死进程的堆栈时使用。请参阅 *_UNW_createContext*(3X) 、 *_UNW_createContextForSelf*(3X) 和在上面的《ACCESSING UNWIND INFORMATION》部分中的“回调”小节。

初始化

在 **_Unwind_Context** 处于 *Init* 状态时，客户端的目标是将给定时间点的处理器状态的快照放置在 **_Unwind_Context** 中。快照由一组请求的寄存器值组成。拍摄快照时必须刷新寄存器堆栈引擎 (RSE)。请参阅处理器规范中的 **flushrs** 指令。

当 **_Unwind_Context** 处于 *Init* 状态时，仅允许初始化。

总是允许写入通用寄存器 1-31、浮点寄存器、判定寄存器、分支寄存器、“当前帧标记” (CFM)、“指令指针” (IP)、和在 **_UNW_AR_RSC**、**_UNW_AR_BSP**、**_UNW_AR_BSPSTORE**、**_UNW_AR_RNAT**、**_UNW_AR_CCV**、**_UNW_AR_UNAT**、**_UNW_AR_FPSR**、**_UNW_AR_ITC**、**_UNW_AR_PFS** 和 **_UNW_AR_LC** 组中的应用程序寄存器。对特定寄存器的写入可以验证 **_Unwind_Context** 中的值。一旦当前帧标记 (CFM) 的值有效，允许对从 GR32 到 GR32 + **CFM.sof** 范围内通用寄存器进行写入（目的是为了在快照过程的 RSE 帧中初始化一个基于 Itanium 系统的 RSE 堆栈通用寄存器）。对 CFM 进行的设置使从 GR32 到 GR127A 范围内的通用寄存器无效。

如上所述，为了使后来对 **_UNW_step()** 的调用成功，必须初始化（因此有效）下列值：

- 指令指针 (IP)。请参阅 *_UNW_setIP*(3X)
- 在 BR0 到 BR7 范围内的保留寄存器和临时分支寄存器。
- 保留寄存器和临时判定寄存器（从 PR1 到 PR63）。
- 在 **_UNW_AR_RSC**、**_UNW_AR_BSP**、**_UNW_AR_BSPSTORE**、**_UNW_AR_RNAT** 和 **_UNW_AR_PFS** 组中的应用程序寄存器。
- 从 GR1 到 GR31 范围内的通用寄存器。请注意，许多通用寄存器都被视为“临时”通用寄存器。如果客户端没有初始化临时通用寄存器，随后对 **_UNW_step()** 的调用仍可能（但不保证）成功，因为大多数过程都不在临时寄存器中保留值。在《Itanium Processor Family Software Conventions and Runtime Architecture》的表 11-13 中定义的通用辗转开解描述符允许过程在临时寄存器中保留值，为了完全，甚至要求初始化临时通用寄存器。对于临时分支、判定和浮点寄存器同样适用。
- 保留寄存器和临时浮点寄存器。
- 当前帧标记 (CFM)。请参阅 *_UNW_setCFM*(3X) 。
- 当前帧标记定义了快照过程的帧中的堆栈通用寄存器。请参阅前面的项目符号和 *_UNW_setCFM*(3X) 。

步进

一旦完成 **_Unwind_Context** 的初始化，则它就代表快照过程的处理器状态。客户端可以调用 **_UNW_step()**，修改 **_Unwind_Context** 以表示 前置对象处理器状态。请参阅 *_UNW_step*(3X) 。

查询

在所有状态中都允许客户端查询寄存器值。然而，仅保证返回值在某些操作后才有效，如下所描述。读取一个未初始化或标记为无效的寄存器会返回零值（浮点寄存器为 0.0，判定寄存器为来自 **_UNW_Boolean** 的

_UNW_FALSE ）并将 **_Unwind_Context** 的 **AlertCode** 设置为 **_UNW_QUERY_INVALID_ERROR** 。请参阅“错误条件和恢复”。通过已记录的范围外面的数字访问通用、浮点、分支、应用或判定寄存器会返回零值，并设置 **_Unwind_Context** 的警报代码为 **_UNW_QUERY_RANGE_ERROR** 。

下面的列表指示了 API 调用特定寄存器值后哪些是有效的。这个“有效性列表”不包括初始化阶段（即，当堆栈辗转开解库处于 *Init* 状态时），在该阶段中，仅已由客户端初始化的那些值是有效的。寄存器按照它们在《Itanium Processor Family Software Conventions and Runtime Architecture》的第 5 章中定义的寄存器类列出。

constant GR0
: 值始终有效。

special GR1 (global pointer)
: 值在从 **_UNW_step** 和 **_UNW_currentContext** 成功返回后有效。

scratch GR2-GR3, GR8-11 and GR14-GR31
: 值仅在 **_UNW_step()** 跨越用户空间中断（例如信号处理程序）后有效。

preserved GR4-GR7
: 值在从 **_UNW_step** 和 **_UNW_currentContext** 成功返回后有效。

special GR12 (stack pointer)
: 值在从 **_UNW_step** 和 **_UNW_currentContext** 成功返回后有效。

special GR13 (thread pointer)
: 值在从 **_UNW_step** 和 **_UNW_currentContext** 成功返回后有效。

stacked GR32-127 (automatic RSE registers)
: 值在从 **_UNW_step** 和 **_UNW_currentContext** 成功返回后有效。注释：仅在当前帧中存在的那些值（如 CFM 值所描述的）有效。当查询当前帧的外部值时会设置警报代码 **_UNW_QUERY_RANGE_ERROR** 。

constant FR0 and FR1
: 值始终有效。

preserved FR2-FR5 and FR16-FR31
: 值在从 **_UNW_step** 和 **_UNW_currentContext** 成功返回后有效。

scratch FR6-FR15 and FR32-FR127
: 值仅在处于 *User_Interrupted_Frame* 状态时 **_UNW_step()** 跨越用户空间中断（例如信号处理程序）后有效。

constant P0
: 值始终有效。

preserved PR1-PR5 and PR16-PR63
: 值在从 **_UNW_step** 和 **_UNW_currentContext** 成功返回后有效。

scratch PR6-PR15

值仅在处于 *User_Interrupted_Frame* 状态时 **_UNW_step()** 跨越用户空间中断（例如信号处理程序）后有效。

scratch BR0 (return link)

值仅在 **_UNW_step()** 跨越用户空间中断（例如信号处理程序）后有效。

不要试图由 BR0 确定前置对象的指令指针。调用 **_UNW_step()** 以获取前置对象帧的 **_Unwind_Context** ，并使用 **_UNW_getIP** 。

preserved BR1-BR5

值在从 **_UNW_step** 和 **_UNW_currentContext** 成功返回后有效。

scratch BR6-BR7

值仅在处于 *User_Interrupted_Frame* 状态时 **_UNW_step()** 跨越用户空间中断（例如信号处理程序）后有效。

application register FPSR

值在从 **_UNW_step** 和 **_UNW_currentContext** 成功返回后有效。

自动应用程序寄存器 **_UNW_AR_RNAT**

值在从 **_UNW_step** 和 **_UNW_currentContext** 成功返回后有效。

保留的应用程序寄存器 **_UNW_AR_UNAT**

值在从 **_UNW_step** 和 **_UNW_currentContext** 成功返回后有效。

特殊应用程序寄存器 **_UNW_AR_PFS**

值在从 **_UNW_step** 和 **_UNW_currentContext** 成功返回后有效。

只读应用程序寄存器 **_UNW_AR_BSP**

值在从 **_UNW_step** 和 **_UNW_currentContext** 成功返回后有效。

特殊应用程序寄存器 **_UNW_AR_BSPSTORE**

值在从 **_UNW_step** 和 **_UNW_currentContext** 成功返回后有效。

应用程序寄存器 **_UNW_AR_RSC**

_UNW_AR_RSC 应用程序寄存器是一个特例。在 **_UNW_step()** 中它标记为有效但并不更新，因为它的值仅为临时和只读的。

保留的应用程序寄存器 **_UNW_AR_LC**

值在从 **_UNW_step** 和 **_UNW_currentContext** 成功返回后有效。

临时应用程序寄存器 **_UNW_AR_CCV**

值仅在 **_UNW_step()** 跨越用户空间中断（例如信号处理程序）后有效。

Convenience value CFM

值在从 **_UNW_step** 和 **_UNW_currentContext** 成功返回后有效。提醒：CFM（当前帧标记）在构成上不可见。对于这个值，堆栈辗转开解库使用与应用程序寄存器 **AR.PFS** 相同的寄存器布局。

为重新初始化清除

客户端可以在除 *Bad* 、 *Start* 和 *Stop* 外的任何状态中调用 **_UNW_clear()** 。 **_UNW_clear** 成功完成时，**_Unwind_Context** 对象会处于“刚构建”的条件。在 **_Unwind_Context** 对象中的所有寄存器值都是无效的，并且它处于 *Init* 状态。

错误条件和恢复

在初始化 **_Unwind_Context** 时检查范围错误，或在 **_Unwind_Context** 对象中执行几个操作后对错误条件进行状态检查，为了客户端可以使用一种便捷的方法，提供了两个 API 函数： **_UNW_getAlertCode** 和 **_UNW_clearAlertCode** 。

因为最后警报代码的清除， **_UNW_getAlertCode()** 允许客户端获取最近的“客户端需要知道” **_UNW_ReturnCode** ，它由接口函数返回（或在查询函数中“遇到”）。

“客户端需要知道”的返回代码是在 **_UNW_ReturnCode** 枚举中的所有枚举成员，但 **_UNW_OK** 除外。

_UNW_clearAlertCode 、 **_UNW_clear** 、 **_UNW_createContextForSelf()** 和 **_UNW_createContext()** 各自重置警报代码为 **_UNW_OK** ，尽管 **_UNW_clear** 、 **_UNW_createContextForSelf()** 和 **_UNW_createContext()** 可以在低内存条件下将警报代码到 **_UNW_MEMORY_ALLOCATION_ERROR** 。

低内存条件

大多数情况下， **_Unwind_Context** 的失败构成会使客户端有足够的 **_Unwind_Context** 对象，以支持调用 **_UNW_getAlertCode()** 函数，这样将在失败构成的情况下返回 **_UNW_MEMORY_ALLOCATION_ERROR** 。如构成成功， **_UNW_getAlertCode()** ，将返回 **_UNW_OK** 。有时可通过将 **_Unwind_Context** 指针设置为 **NULL** 来与失败构成通信。如果 **_Unwind_Context** 指针是 **NULL** ，则对堆栈辗转开解库接口函数的后继调用将返回 **_UNW_MEMORY_ALLOCATION_ERROR** 。

操作中的内存分配失败（例如 **_UWN_step()** ）会返回 **_UNW_MEMORY_ALLOCATION_ERROR** 返回代码，并设置 **_Unwind_Context** 警报代码。 **_Unwind_Context** 的其余状态未定义。

返回代码语义

枚举 **_UNW_ReturnCode** 具有一些嵌入语义。 **_UNW_OK** 为 0 值。为堆栈边界条件分配负值，该堆栈边界条件指示堆栈辗转开解库不能通过其步进的堆栈帧。例如， **_UNW_STEP_BOTTOM** 和 **_UNW_STEP_KERNEL_SAVE_STATE** 为负值。枚举中所有其他返回值为正。 HP 保留将更多返回代码添加到枚举的权利。出于兼容性的原因，不能进行任何删除，分配到现有返回代码的值在将来的版本中也不能更改。

另请参阅

_UNW_createContextForSelf(3X)、 _UNW_currentContext(3X)、 _UNW_getGR(3X)、 U_STACK_TRACE(3X)。

《Itanium Processor Family Runtime Architecture Supplement》

《Itanium Processor Family Software Conventions and Runtime Architecture》

《Processor-Specific ELF Supplement for IA-64》

名称

values - 与计算机相关的值

概要

#include <values.h>

说明

该文件包括一组明示常数，这些常数是有条件地为特定处理器体系结构定义的。

该模型假定整数使用二进制表示形式（1 或 2 的补数），其中符号位由高次位的值表示。

BITS(*type*) 指定类型的位数（例如，int）。

HIBITS 仅有高次位集的短整数的值（大多数实现为 0x8000）。

HIBITL 仅有高次位集的长整数的值（大多数实现为 0x80000000）。

HIBITI 仅有高次位集的常规整数的值（通常与 **HIBITS** 或 **HIBITL** 相同）。

MAXSHORT 有符号短整数的最大值（大多数实现为 0x7FFF ≡ 32767）。

MAXLONG 有符号长整数的最大值（大多数实现为 0x7FFFFFFF ≡ 2147483647）。

MAXINT 有符号常规整数的最大值（通常与 **MAXSHORT** 或 **MAXLONG** 相同）。

MAXFLOAT, LN_MAXFLOAT
单精度浮点数的最大值及其自然对数。

MAXDOUBLE, LN_MAXDOUBLE
双精度浮点数的最大值及其自然对数。

MINFLOAT, LN_MINFLOAT
单精度浮点数的最小正值及其自然对数。

MINDOUBLE, LN_MINDOUBLE
双精度浮点数的最小正值及其自然对数。

FSIGNIF 单精度浮点数的尾数中重要位的数量。

DSIGNIF 双精度浮点数的尾数中重要位的数量。

文件

/usr/include/values.h

另请参阅

intro(3)、math(5)。

符合的标准

<values.h>： XPG2

名称

varargs - 处理变量参数列表

概要

#include <varargs.h>

va_alist

va_dcl

void va_start(pvar)
va_list pvar;

type **va_arg(pvar,** *type***)**
va_list pvar;

void va_end(pvar)
va_list pvar;

说明

这组宏使程序员能够编写可以接受变量参数列表的可移植过程。拥有变量参数列表（例如 **printf()**），但不使用 **varargs** 的例行程序，由于不同计算机使用不同的参数传递惯例（请参阅 *printf*(3S)），因此它们本质上是不可移植的。

va_alist 在函数头中用作参数列表。

va_dcl 是对 **va_alist** 的声明。 **va_dcl** 之后不跟分号。

va_list 是一个类型，定义用于遍历列表的变量。

调用 **va_start** 将 *pvar* 初始化到列表的起始位置。 *argN* 的类型应该和位于参数列表中变量部分以前的函数的参数相同。

va_arg 返回 **pvar** 指向的列表中的下一个参数。 *type* 是该参数应采用的类型。不同的类型可以混用，但是因为无法在运行时确定，所以由例行程序确定参数的类型。

va_end 用于清除。

多次遍历（每次由 **va_start** ... **va_end** 括起来）是可能的。

注释：提供 **<varargs.h>** 头文件是为了与 ANSI 前的编译程序和早期的 HP C/HP-UX 版本兼容。它由 **<stdarg.h>** 所取代，后者包含所有 **varargs** 宏。

举例

下面的示例显示了对 **execl()** 的一种可能实现（请参阅 *exec*(2)）：

```
#include <varargs.h>
#define MAXARGS     100

/*  execl is called by
```

```
    execl(file, arg1, arg2, ..., (char *)0);
*/
execl(va_alist)
va_dcl
{
    va_list ap;
    char *file;
    char *args[MAXARGS];
    int argno = 0;

    va_start(ap);
    file = va_arg(ap, char *);
    while ((args[argno++] = va_arg(ap, char *)) != (char *)0);

    va_end(ap);
    return execv(file, args);
}
```

下一个示例说明接受变量参数的函数如何将这些参数传递到其他函数。要完成这些，第一个接收了变量参数列表的例行程序（在此示例中为 log_errors() ）必须将通过调用 **va_start()** 得到的地址指针传递到任何需要访问相同变量参数列表的后续调用中。所有接收此地址指针的例行程序（在此示例中为 v_print_log() ），仅需要使用 **va_arg()** 来访问原始变量参数列表，就好像它们也是要被传递变量参数的原始例行程序一样。

在此示例中，可以想象有一系列其他例行程序（例如 **log_warning()** 和 **log_message()** ）也调用 **v_print_log()** 函数。

```
#include <stdio.h>
#include <varargs.h>
#include <unistd.h>

int error_count;

/* VARARGS4 -- for lint */
int
log_errors(log_fp, func_name, err_num, msg_fmt, va_alist)
FILE *log_fp;
char *func_name;
int err_num;
char *msg_fmt;
va_dcl
{
```

```
    va_list ap;

    /* Print error header information */
    (void) fprintf(log_fp, "\nERROR in process %d\n", getpid());
    (void) fprintf(log_fp, "  function \"%s\": ", func_name);
    switch(err_num)
    {
       case ILLEGAL_OPTION:
          (void) fprintf(log_fp, "illegal option\n");
          break;
       case CANNOT_PARSE:
          (void) fprintf(log_fp, "cannot parse input file\n");
          break;
       ...
    }

   /*
    * Get pointer to first variable argument so that we can
    * pass it on to v_print_log().  We do this so that
    * v_print_log() can access the variable arguments passed
    * to this routine.
    */
    va_start(ap);

    v_print_log(log_fp, msg_fmt, ap);

    va_end(ap);
}

/* VARARGS2 -- for lint */
int
v_print_log(log_fp, fmt, ap)
FILE *log_fp;
char *fmt;
va_list ap;
{

    /*
     * If "%Y" is the first two characters in the format string,
```

```
 * a second file pointer has been passed in to print general
 * message information to.  The rest of the format string is
 * a standard  printf(3S) format string.
 */
if ((*fmt == '%') && (*(fmt + 1) == 'Y'))
{
   FILE *other_fp;

   fmt += 2;

   other_fp = (FILE *) va_arg(ap, char *);
   if (other_fp != (FILE *) NULL)
   {
      /*
       * Print general message information to additional stream.
       */
      (void) vfprintf(other_fp, fmt, ap);
      (void) fflush(other_fp);
   }
}

/*
 * Now print it to the log file.
 */
(void) vfprintf(log_fp, fmt, ap);
}
```

警告

由调用例行程序来指定有参数的数量，因为不是总可以通过堆栈帧来确定这一数量的。例如，向 **execl()** 传入一个零指针来标识列表的结束。 **printf()** 能够确定以此格式存在的参数的数量。

为 **va_arg** 指定第二个 **char** 、 **short** 或 **float** 参数是不可移植的，因为调用函数看到的这些参数不是 **char** 、 **short** 或 **float** 。在将这些参数传递到函数之前，C 将 **char** 和 **short** 参数转换为 **int** ，并且将 **float** 参数转换为 **double**。

另请参阅

exec(2)、 vprintf(3S)、 stdarg(5)。

符合的标准

va_alist：AES、SVID2、SVID3、XPG2、XPG3、XPG4

va_arg：SVID2、SVID3、XPG2、XPG3、XPG4

va_dcl：SVID2、SVID3、XPG2、XPG3、XPG4

va_end： SVID2、SVID3、XPG2、XPG3、XPG4

va_list： SVID2、SVID3、XPG2、XPG3、XPG4

va_start： SVID2、SVID3、XPG2、XPG3、XPG4

<varargs.h>： AES、SVID3、XPG2、XPG3、XPG4

名称

vps_ceiling - 系统可选的最大页面大小（以 KB 为单位）

值

缺省值

16 KB

允许值

最小值： **4** KB

最大值：**4194304** KB

说明

转换后备缓冲区 (TLB) 是虚拟内存的微处理器功能，它对最近使用的物理地址与虚拟地址之间的转换进行缓存，希望这些转换可能很快被再次使用。 TLB 以程序中地址引用的空间和时间的位置要素为基础。以前，TLB 的管理完全是在硬件中进行的，目的是为了实现速度优化，但却同时失去了软件实现的灵活性。例如，一些容易更改的算法或表的实现。

近年来，与纯粹的硬件速度相比， TLB 软件实现的灵活性更受重视。特别是，将物理帧（在硬件中，其大小是固定的）以逻辑方式分组为“superpage”和“largepage”，这一思想的代表是软件 TLB 算法，在该算法中，多个物理帧使用一个基本地址转换；这就显著地减少了因页面缺失而造成的丢失周期（假设空间和时间位置都合理）。例如，假设一个科学应用程序使用一个数组，数组中的每个元素需要 1 KB 的内存。使用常见的 4 KB 物理帧并依次引用数组将导致缺页，需要从磁盘或交换区将页面读入内存，而且每 5 个元素就需要载入带有帧基本地址转换的 TLB。

如果用户应用程序没有使用 **chatr** 命令为程序的文本段和数据段指定页面大小，则内核将根据系统配置和对象大小自动选择一个页面大小。然后，比较自动选择的大小与可调参数 **vps_ceiling** 定义的最大页面大小，如果自动选择的大小更大，则使用 **vps_ceiling** 的值。然后，比较选定的值与通过 **vps_pagesize** 设置的最小页面大小，并使用两个值中的较大者。

更改此可调参数的人员

任何用户。

更改限制

此可调参数的更改对此后任何的物理内存分配均有效。它不影响任何已经被分配的物理内存。

应该何时增加此可调参数的值

当系统上的进程以常规方式访问自已的文本和数据时，可以增加这个可调参数的值，而且一定范围内的数据大于当前值。例如，如果该可调参数设置为 16 KB，但是系统上几乎每个进程都反复使用四或五个不同的 256 KB 数据集，那么，将该可调参数的值增加为 256 将减少进程的缺页，因为现在一个 256 KB 转换替换了原来的 16 个 16 KB 页面地址转换。

一般的系统行为并不会表现出内存访问上的一致性，而且该可调参数的最优值不易确定，因此该可变参数仅表示内核试探性的上限值，而且不能改变实际的系统行为。

增加此值的负面影响

如果 **vps_pagesize** 也增加，或者内核试探性选择一个较大的值，则内存分配将需要更大的相邻页分组。这可能导致意外行为。例如，在缺省值下，当程序从磁盘读入最后 4 KB 的代码时，将需要 16 KB 的连续物理内存，而且还需要建立相应的虚拟转换，尽管实际上只使用 4 KB 的数据。请考虑参数设置为最大值的情况，这种情况下，需要为程序所使用的每个新虚拟页分配 64 MB 的连续物理内存，即使实际上仅使用 4 KB 也是如此。这种情况下不仅浪费了物理内存，而且，由于可能没有可用的连续物理内存帧，进程可能暂停执行，等待为其分配实际需要的内存量。

因此，最好在准确地了解系统的内存使用情况之后再决定是否增加该可调参数的值。通常，更好的实践惯例是，对于已知应用程序按每个应用程序的基础增加可变页面大小，例如，扫描大量数据的数据库只有一个缺页。

现代的体系结构可支持非常大的页面（Itanium 最大可支持 4 GB， PA-RISC 最大可支持 1 GB）。将 **vps_ceiling** 设置为非常大的页面大小（大于 64 KB）应该特别谨慎，因为它可能导致过多的内存消耗，从而使计算机上可用的空闲内存迅速耗尽。

应该何时降低此可调参数的值

如果由于物理内存碎片等待被转换为连续的内存块，而禁止运行使用较小内存的进程，或者内存的整体系统使用表明空间位置不好（内存的虚拟访问不是相邻的），而导致产生废弃的物理内存帧，则应该减小可调参数的值。

降低此值的负面影响

对于数据库等应用程序，将工作集载入内存时，将遇到更多的缺页，但是通过使用相应的应用程序和 **chatr** 可以处理这一问题。

应该同时更改的其他可调参数

当该可调参数的值为内核试探性范围的下限值时，还应考虑 **vps_pagesize** 。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

vps_ceiling 由 HP 开发。

另请参阅

vps_pagesize(5)。

名称

vps_chatr_ceiling - 用户可选的最大页面大小（以 KB 为单位）

值

缺省值

1048576 (KB)

允许值

最小值： **4** (KB)

最大值： **4194304** (KB)

说明

转换后备缓冲区 (TLB) 是虚拟内存的微处理器功能，它对最近使用的物理地址与虚拟地址之间的转换进行缓存，希望这些转换可能很快再次需要。 TLB 以程序中地址引用的空间和时间的位置要素为基础。以前，TLB 的管理完全是在硬件中进行的，目的是为了实现速度优化，但却同时失去了软件实现的灵活性。例如，一些容易改变的算法或表的实现。

近年来，与纯粹的硬件速度相比， TLB 软件实现的灵活性更受重视。特别是，将物理帧（在硬件中，其大小是固定的）以逻辑方式分组为"超级页"或"大型页"，这一思想的代表是软件 TLB 算法，在该算法中，多个物理帧使用一个基本地址转换；这就显著地减少了因页面缺失而造成的丢失周期（假设空间和时间位置都合理）。例如，假设一个科学应用程序使用一个数组，数组中的每个元素需要 1 KB 的内存。使用常见的 4 KB 物理帧并依次引用数组将导致缺页，需要从磁盘或交换区将页面读入内存，而且每 5 个元素就需要载入带有帧基本地址转换的 TLB。

vps_chatr_ceiling 可调参数可以通过 **chatr** 或任何编程接口或其他机制，设置用户指定的文件上的虚拟页面大小的上限值（编程接口指可从程序调用的接口）。

最初，创建此可调参数是为了设置虚拟页面大小的上限值，而虚拟页面大小是用户通过二进制程序使用 **chatr** 命令设置的。因此该参数名为 **vps_chatr_ceiling** （其中"vps"指 可变的页面大小）。

更改此可调参数的人员

任何用户。

更改限制

此可调参数的更改对此后任何的物理内存分配均有效。它不影响任何已经被分配的物理内存。

应该何时增加此可调参数的值

当用户应用程序，或通常使用大型内存集的系统（如数据库）所需要的页面大于当前值所允许的页面时，应增加该可调参数的值。

增加此值的负面影响

负面影响取决于系统中实际的内存使用情况，以及是否不加选择（即并未基于性能考虑）地使用较大页面。

在第一种情况下，当应用程序以稀疏模式访问内存，或者通常使用很小的工作集时，错误地指出应用程序使用了较大的页面大小（例如，512 MB 或更大）。例如，一个应用程序使用 Shell 脚本，但此脚本总共仅需要 64 KB 的内存；或者，一个科学备用数组分析程序需要处理大型数据集，但是它实际只需要处理小部分的数据，剩下的数据可以进行交换，或者甚至无需分配。为应用程序设置此值将造成一些物理内存帧的浪费，因为如果核心内存中存在任何虚拟的大型页面，则必须将其全部加载。这可能导致不必要的页面调出和调入现象，即过多的磁盘活动，进而造成性能的下降。

在第二种情况下，如果多个用户均在无适当的理由的情况下，为其应用程序申请使用较大的页面大小，让此可调参数保持低值可以最大限度地降低其对系统其他部分的性能影响。

现代的体系结构可支持非常大的页面大小（IA 最大可支持 4 GB， PA-RISC 最大可支持 1 GB）。将 vps_chatr_ceiling 设置为非常高的值（大于 64 KB）应该特别谨慎。它可能造成过多的内存消耗，从而使系统上可用的空闲内存迅速耗尽。

应该何时降低此可调参数的值

如果没有用户程序真正需要将当前的可调参数值调整为大页面大小，则应该降低此可调参数的值，以最大限度减少出现错误，或者恶意用户申请不必要的较大页面大小造成物理帧浪费的机会。

降低此值的负面影响

一个负面影响在于，将使用较小的页面大小来运行程序。此外，系统的某些系统页面也将使用较小的页面大小。

应该同时更改的其他可调参数

无。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

vps_chatr_ceiling 由 HP 开发。

另请参阅

chatr(1)。

名称

vps_pagesize - 系统选择的最小页面大小（以 KB 为单位）

值

缺省值

16 (KB)

允许值

最小值： **4** (KB)

最大值： **4194304** (KB)

说明

转换后备缓冲区 (TLB) 是虚拟内存的微处理器功能，它对最近使用的物理地址与虚拟地址之间的转换进行缓存，希望可能很快再次需要这些转换。 TLB 以程序中地址引用的空间和时间的位置要素为基础。以前，TLB 的管理完全是在硬件中进行的，目的是为了实现速度优化，但却同时失去了软件实现的灵活性。例如，一些容易更改的算法或表的实现。

近年来，与纯粹的硬件速度相比， TLB 软件实现的灵活性更受重视。特别是，将物理帧（在硬件中，其大小是固定的）以逻辑形式分组为“超级页”和“大型页”，这一思想的代表是软件 TLB 算法，在该算法中，多个物理帧使用一个基本地址转换；这就显著地减少了因页面缺失而造成的丢失周期（假设空间和时间位置都合理）。例如，假设一个科学应用程序使用一个数组，数组中的每个元素需要 1 KB 的内存。使用常见的 4 KB 物理帧并依次引用数组将导致缺页，需要从磁盘或交换区将页面读入内存，而且每 5 个元素就需要载入带有帧基本地址转换的 TLB。

如果用户应用程序没有使用 **chatr** 命令为程序的文本段和数据段指定页面大小，则内核将根据系统配置和对象大小自动选择一个页面大小。然后，比较自动选择的大小与 **vps_ceiling** 可调参数定义的最大页面大小，如果自动选择的大小更大，则使用 **vps_ceiling** 的值。然后，比较选定的值与通过 **vsp_pagesize** 设置的最小页面大小，并使用两个值中的较大者。

更改此可调参数的人员

任何用户。

更改限制

此可调参数的更改对此后的物理内存分配有效。使用中的物理内存将不受影响。

应该何时增加此可调参数的值

当系统上的进程以常规方式，通过大于当前值的数据范围访问自已的文本和数据时，可以增加这个可调参数的值。例如，如果该可调参数设置为 16 KB，但是系统上几乎每个进程都反复使用四或五个不同的 256 KB 数据集，那么，将该可调参数的值增加为 256 将减少进程的缺页，因为现在一个 256 KB 转换替换了原来的 16 个 16 KB 页面地址转换。

一般的系统行为并不会表现出内存访问上的一致性，而且该可调参数的最优值不易确定，因此该可变参数仅表示内核试探性的较低值，而且可能不会改变实际的系统行为。

增加此值的负面影响

内存分配将需要更大的连续页分组，因为内核试探性不选择较大的值。

要求较大的虚拟页可能导致产生意外的系统行为。特别是，当许多个使用较小或零碎的数据/代码集的进程处于活动状态时，情况尤其如此。被应用程序引用的每一个虚拟页，无论页内的实际使用情况如何，都要求整页的连续物理内存帧工作。例如，将不能交换调出大型虚拟页面的一半。许多个连续的帧并非总是可能的，而且可能导致等待内存分配，尽管这些内存分配并非不可或缺的。此外，在这种情况下，物理内存帧的浪费还可能导致更多地使用交换，从而进一步降低系统性能。

现代的体系结构可支持非常大的页面大小（ Itanium® 最大可支持 4 GB，PA-RISC 最大可支持 1 GB）。将此值设置为非常高的值（大于 64 KB）可能造成过多的内存消耗，从而使系统上可用的空闲内存迅速耗尽。

应该何时降低此可调参数的值

如果由于物理内存碎片等待被转换为连续的内存块，而禁止运行使用较小内存的进程，或者内存的整体系统使用表明空间位置不好（内存的虚拟访问不是相邻的），而导致产生废弃的物理内存帧，则应该减小可调参数的值。

降低此值的负面影响

如果同时也减小 **vps_ceiling** 的值，则使用大型数据集的应用程序（如数据库）将因为页面缺失的增加而性能降低。适当的应用程序使用 **chatr** 命令可以解决这一问题。如果未修改 **vps_ceiling** ，则负面影响可以降到最小，因为内核可以在更大范围内为未被 **chatr** 命令修改的每个应用程序选择合适的页面大小。

应该同时更改的其他可调参数

当该可调参数的值为内核试探性范围的下限值时，还应考虑 **vsp_ceiling** 。

警告

所有 HP-UX 内核可调参数都是针对于特定发行版的。在将来的 HP-UX 发行版中可能会删除此参数，或改变其含义。

安装来自 HP 或其他供应商的可选内核软件可能会导致更改可调参数的值。安装之后，某些可调参数可能不再是缺省值或建议的值。有关安装对可调参数值产生的影响的信息，请查阅所安装内核软件的文档。有关出厂安装在用户系统上的可选内核软件的信息，请参阅 **http://docs.hp.com** 上的《HP-UX Release Notes》。

作者

vps_pagesize 由 HP 开发。

另请参阅

vps_ceiling(5)。

名称

xferlog - FTP 服务器日志文件

概要

/var/adm/syslog/xferlog

说明

xferlog 文件包含来自 FTP 服务器守护程序 **ftpd** （请参阅 *ftpd*(1M) ）的日志记录信息。该文件位于 **/var/adm/syslog** 。每个服务器条目由下列形式的单独一行构成，所有字段由空格进行分隔。

current-time transfer-time remote-host file-size filename transfer-type special-action-flag direction access-mode username service-name authentication-method authenticated-user-id completion-status

current-time 以“DDD MMM dd hh:mm:ss YYYY”形式表示的当前本地时间。其中 DDD 表示星期几，MMM 表示月份，dd 是日期， hh 表示小时，mm 表示分钟，ss 表示秒，YYYY 表示年份。

transfer-time 传输所需全部时间，以秒为单位。

remote-host 远程主机名。

file-size 传输文件的大小，以字节为单位。

filename 传输的文件的名称。

transfer-type 指示传输类型的单个字符。可以是下列各项之一：

- **a** ASCII 传输
- **b** 二进制传输

special-action-flag 一个或多个单个字符标志指示要采取的任何特殊操作。可以是下列各项之一：

- **C** 文件被压缩
- **U** 对文件解压缩
- **T** 文件已由 tar 程序处理
- _ 未采取任何操作

direction 传输的方向。可以是下列各项之一：

- **o** 传出
- **i** 传入

access-mode 用户登录的方法。可以是下列各项之一：

- **a** (anonymous) 是匿名的 guest 用户使用的。
- **g** (guest) 由有口令的 guest 用户使用（请参阅 *ftpaccess*(4) 中的 **guestgroup** 命令）。
- **r** (real) 由经过验证的本地用户使用。

username 本地用户名，或者如果是 guest 的话，则为给定的 ID 字符串。

service-name 被调用的服务的名称，通常是 FTP。

authentication-method 使用的验证方法。可以是下列各项之一：

0 无

1 RFC931 验证

authenticated-user-id 由验证方法返回的用户 ID。如果无法获取所验证用户 ID，则使用 *。

completion-status 这是指示传输状态的单个字符。可以是下列各项之一：

c 完全传输

i 不完全传输

文件

/usr/adm/syslog/xferlog

作者

xferlog 由密苏里州华盛顿大学圣路易斯分校开发。

另请参阅

ftpd(1M)、 ftpaccess(4)、 syslog(3C)。